Implémentez un modèle de scoring

Projet 7 du parcours
Data Scientist

Dernière MàJ
24 août 2023

Raquel Sanchez Pellicer

OPENCLASSROOMS

Prêt à dépenser

Problématique et données

Modélisation

Pipeline déploiement

Analyse data drift

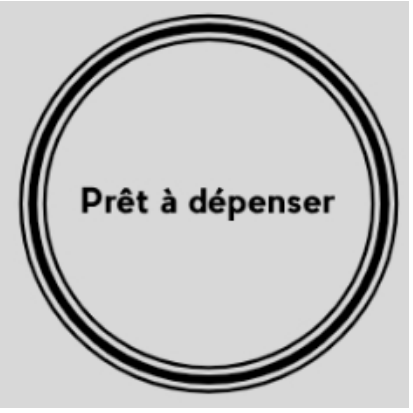
API et Dashboard

Conclusions et perspectives

Annexes

Problématique

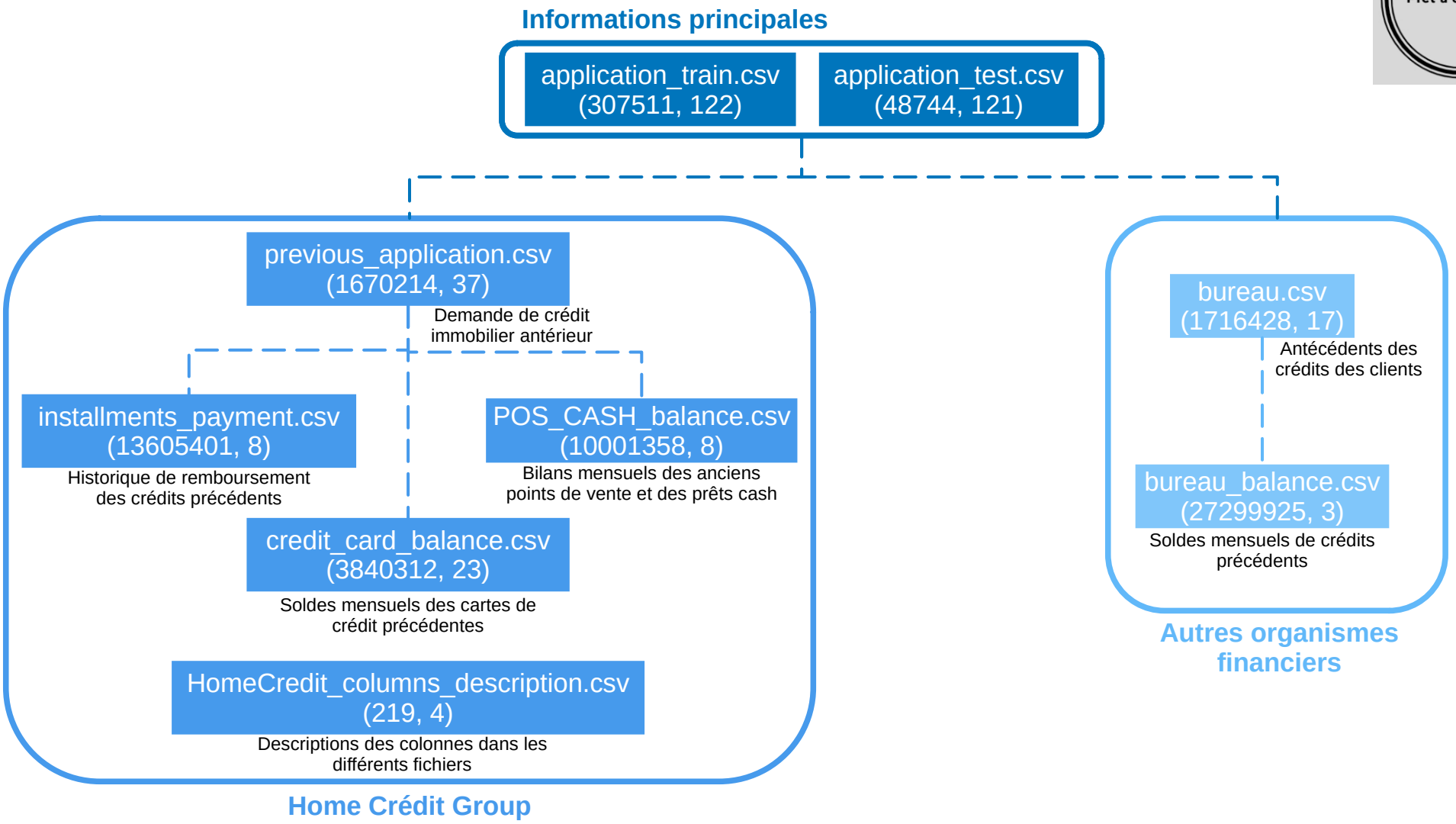
Prêt à dépenser souhaite mettre en place un **outil d'évaluation** du crédit pour calculer la probabilité qu'un client rembourse son prêt, puis **classer la demande** dans la catégorie des crédits **accordés ou refusés**. Elle souhaite donc développer un algorithme de classification basé sur une variété de sources de données (données comportementales, données provenant d'autres institutions financières, etc.)



En outre, les responsables des relations avec la clientèle ont souligné que les clients exigent de plus en plus de transparence dans les décisions d'octroi de crédit. Cette demande de transparence de la part des clients est tout à fait conforme aux valeurs que l'entreprise souhaite incarner.

Prêt à dépenser a donc décidé de **développer un tableau de bord interactif** afin que les responsables de la relation client puissent expliquer les décisions d'octroi de crédit de la manière la plus transparente possible, mais aussi permettre à leurs clients d'accéder et d'explorer facilement leurs informations personnelles.

Données disponibles



Mission

- L'objectif principal est développer un modèle permettant d'**identifier les mauvais payeurs** sur la base des données fournies par Prêt a dépenser.
- Prendre en compte le **coût des erreurs**. Le coût d'une erreur peut être très élevé. Nous devons veiller à minimiser le nombre de mauvais payeurs auxquels le modèle accorderait un crédit, ce qui pourrait entraîner d'énormes pertes financières pour l'organisation.
- **L'interprétabilité** est importante, on doit pouvoir justifier les décision face aux clients.
- Il n'y a **pas de contrainte stricte en matière de temps de latence**, car l'objectif est davantage de prendre la bonne décision que de prendre une décision rapide. Il est acceptable que le modèle prenne quelques secondes pour faire une prédiction.

Prêt à dépenser

Problématique et données

Modélisation

- Préparation données
- Sélection des métriques
- Optimisation hyperparamètres
- Modèle choisi

Pipeline déploiement

Analyse data drift

API et Dashboard

Conclusions et perspectives

Annexes

Processus modélisation

Prêt à dépenser

Préparation des données

Données originaux

Données pour entraîner le modèle

Sélection des métriques

Métriques robustes pour comparaison des performances

Métriques adaptés à la problématique métier

Tests algorithmes

5 algorithmes différents

Optimisation hyperparamètres

Meilleur modèle

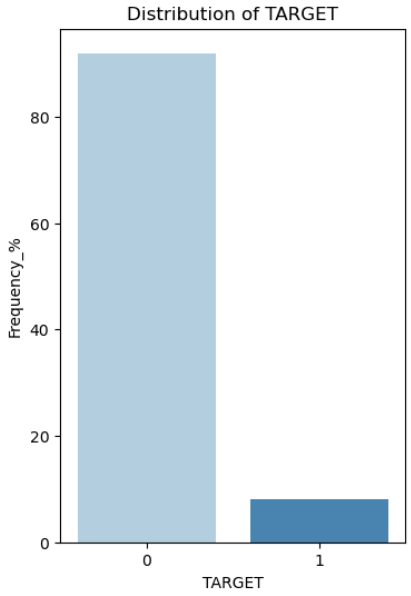
Hyperparamètres et performance

Interprétabilité

Données disponibles

Exploration application_train

Target	Signification	Objectif	Signification
1	Client avec difficultés de paiement	1	Crédit refusé
0	Autres cas	0	Crédit accordé



Préparation des données

Prêt à dépenser

Nettoyage : Types de données, Valeurs aberrantes, uniques, manquants

- Objets Y/N ou Yes/No en int8
- Numérique ordinal en objet
- Réduction pour optimiser la mémoire

- Correction des valeurs aberrantes

- Suppression si nan >68 %
- Imputation quanti : médiane ou 0 ou NaNImputer
- Imputation quali : mode ou XNA

Feat. engineering : Création var. métier et statistiques + Encodage + colinéarités + Merge

- Métier : création manuelle
- Statistiques : min, max…
- Encodage quanti. : StandardScaler
- Encodage quali. : LabelEncoder et get_dummies
- Détails en annexe

- Suppr. Variables correl > 0.7 (coef. Pearson)

Feat. selection : Variables sans info. + Feat. Imp.

- Suppr. Variables 1 seul valeur
- Feat importance

2 fichiers (train, test)

34 variables, les 25 le plus d'imp. LightGBM & Random Forest

Sélection des métriques

Prêt à dépenser

Métriques classiques :

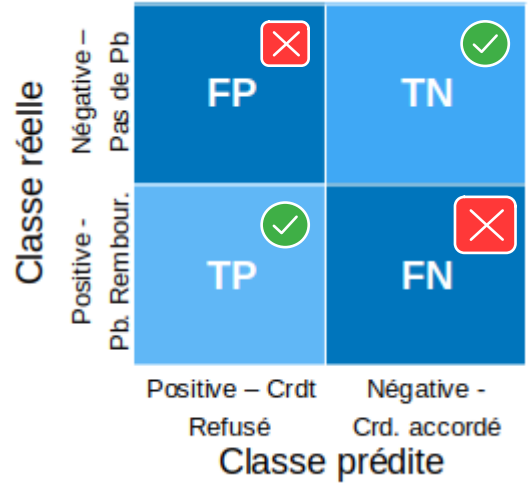
Recall : (rappel) taux de vrais positifs (parmi les vrais positifs combien ont été classées comme positifs)

Precision : combien des prédits comme positifs l'étaient en vrai

F-beta : moyenne harmonique des valeurs de *Recall* et precision, si β > 1 on donne plus d'importance au *Recall* **(F-2 retenu)**

ROC AUC : Aire au dessous de la courbe représentant les valeurs de *Precision* et de *Recall* du modèle en fonction du seuil du score de confiance du modèle.

Average Precision, (AP) : résume la courbe de *Precision - Recall* en une seule valeur. Pour définir le terme, l'AP est la moyenne pondérée des scores de *Precision* obtenus à chaque seuil de la courbe PR, l'augmentation du *Recall* par rapport au seuil précédent étant utilisée comme pondération.

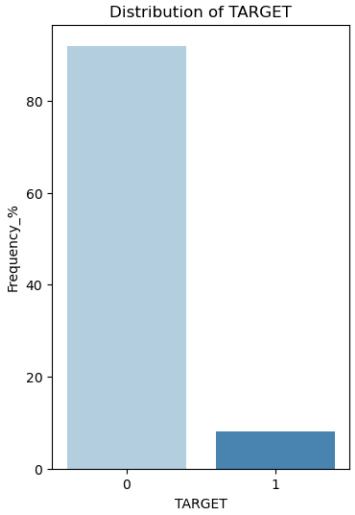


Conception métrique métier:

Business score: minimiser les **FN (-10)** en veillant à la réduction des **FP (-1)** et maximiser les TP.

Test algorithmes, Data imbalance

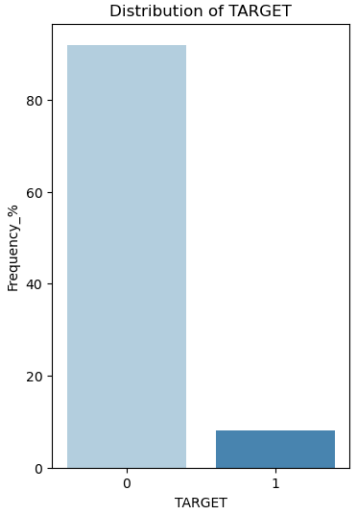
Objectif : identifier la meilleure méthode pour rééquilibrer la target



Test algorithmes, Data imbalance

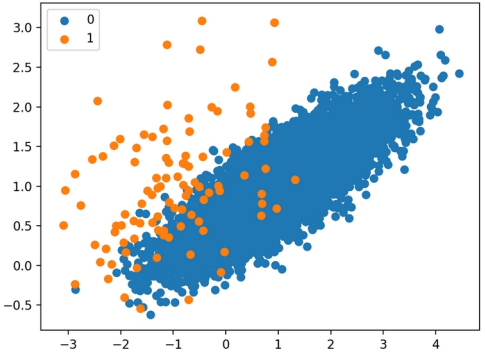
Objectif : identifier la meilleure méthode pour rééquilibrer la target

Comment : entraînement de 5 algorithmes, hyperparamètres par défaut, avec trois méthodes de rééquilibrage différents

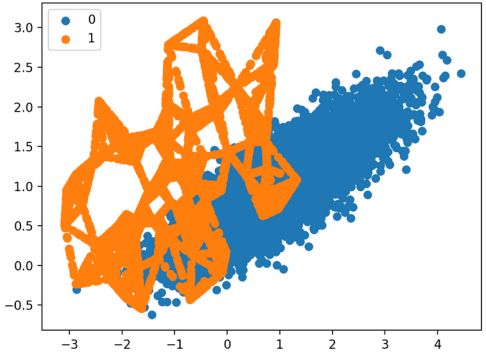


Algorithmes	Méthod
➢ Baseline (Dummy) ➢ Logistic Regression ➢ Random Forest ➢ XG Boost ➢ LightGBM	➔ **SMOTE** (Oversampling) ➔ **SMOTE + RandomUnderSampling** (Oversampling + Undersampling) ➔ **Class_weight** (échantillonnage des classes pondéré en fonction de leur fréquence)

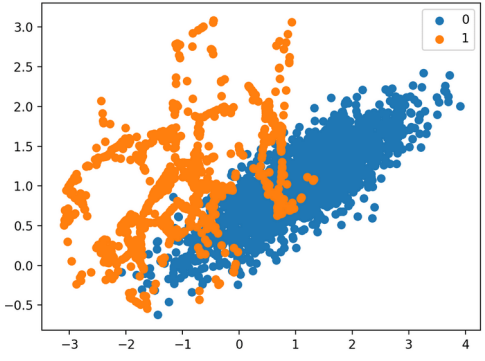
Dataset Original



Dataset après SMOTE



Dataset après SMOTE + RandomUnderSampling



Test algorithmes, Data imbalance

Prêt à dépenser

mlflow 2.5.0 Experiments Models

Experiments

Search Experiments

- hyperparam_tunnig
- data_rebalancing

data_rebalancing Provide Feedback

Experiment ID: 746537749930035109 Artifact Location: file:///home/raquelsp/Documents/Openclassrooms/P7_implementez_modele_scoring/P7_travail/P7_scoring_credit/mlruns/746537749930035109

Description Edit

Metrics

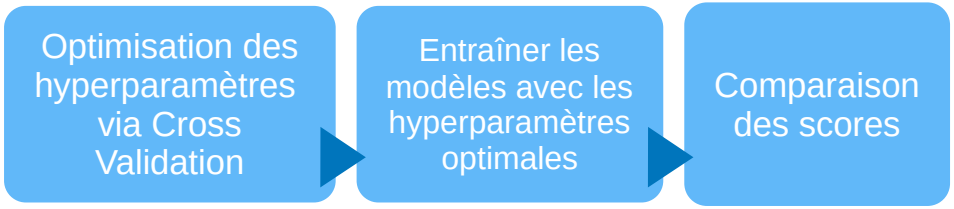
Run Name	Duration	AP_SCORE	Business_score	F2	FN	ROC_AUC
LGBM_defaultParams_classWeight	2.0s	0.266	0.705	0.432	1553	0.771
LGBM_defaultParams_unbalanced	1.9s	0.268	0.682	0.032	4836	0.771
XGBClass_defaultParams_unbalanced	1.5s	0.259	0.687	0.059	4727	0.766
logistic_regression_defaultParams_classWeight	1.3s	0.227	0.685	0.411	1621	0.745
logistic_regression_defaultParams_SMOTE_UNDER	1.4s	0.226	0.686	0.413	1607	0.744

Run Name	Duration	AP_SCORE	Business_score	F2	FN	ROC_AUC
XGBClass_defaultParams_classWeight	1.3s	0.23	0.398	0.322	197	0.725
LGBM_defaultParams_classWeight	2.0s	0.266	0.705	0.432	1553	0.771
logistic_regression_defaultParams_SMOTE_UNDER	1.4s	0.226	0.686	0.413	1607	0.744
logistic_regression_defaultParams_SMOTE	1.4s	0.226	0.686	0.412	1616	0.744
logistic_regression_defaultParams_classWeight	1.3s	0.227	0.685	0.411	1621	0.745

metrics.FN

Run Name	Duration	AP_SCORE	Business_score	F2	FN	ROC_AUC
RForestClass_defaultParams_SMOTE	281ms	0.189	0.712	0.301	3313	0.729
RForestClass_defaultParams_SMOTE_UNDER	267ms	0.189	0.712	0.301	3313	0.729
XGBClass_defaultParams_SMOTE_UNDER	1.4s	0.217	0.712	0.225	3904	0.737
LGBM_defaultParams_SMOTE_UNDER	1.8s	0.205	0.709	0.231	3844	0.736
LGBM_defaultParams_classWeight	2.0s	0.266	0.705	0.432	1553	0.771

Test algorithmes, Optimisation hyperparamètres



Dummy (Baseline)	most frequent
Logistic Regression	Penalty : L1, L2, elasticnet Solver : liblinear
Random forest	max_depth : 5, 10, 50, None max_features : auto, sqrt Criterion : gini, entropy
XGBoost	learning_rate : 0.1, 0.01, 0.05 Gamma : 0, 0.5, 2, 5, Subsample : 0.6, 1.0 max_depth : 4, 6
LightGBM	max_depth : 10, 50, 150 n_estimators : 100, 200 learning_rate : 0.01, 0.01, 1

Test algorithmes, Comparaison des modèles

Prêt à dépenser

mlflow 2.5.0 Experiments Models GitHub Docs

Experiments

Search Experiments

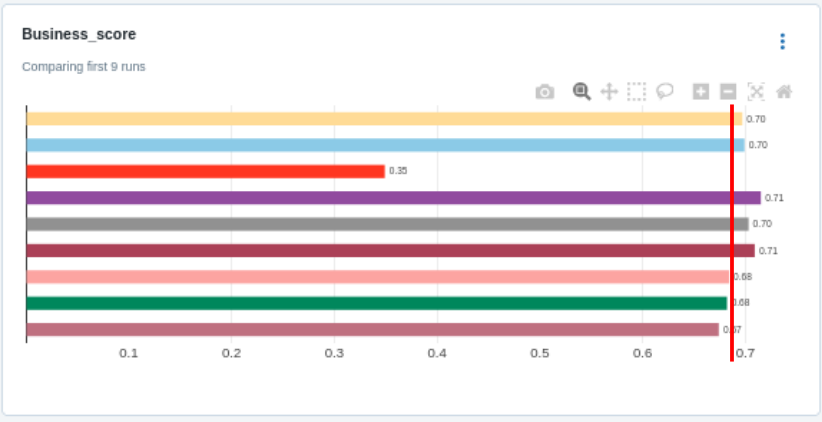
- hyperparam_tunnig
- data_rebalancing

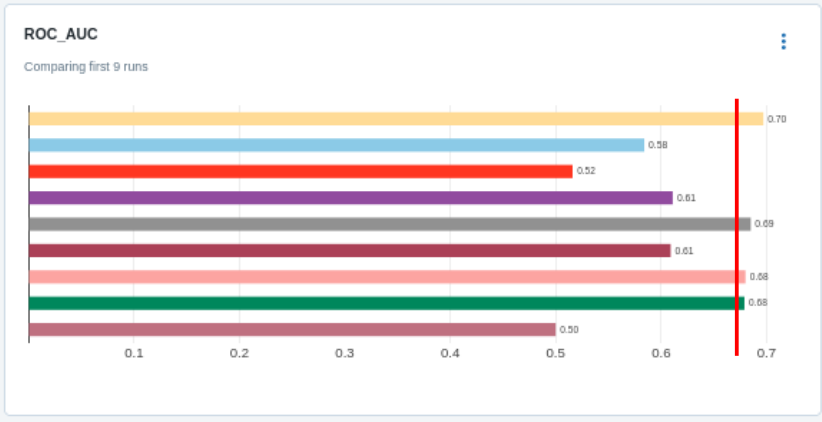
hyperparam_tunnig Provide Feedback

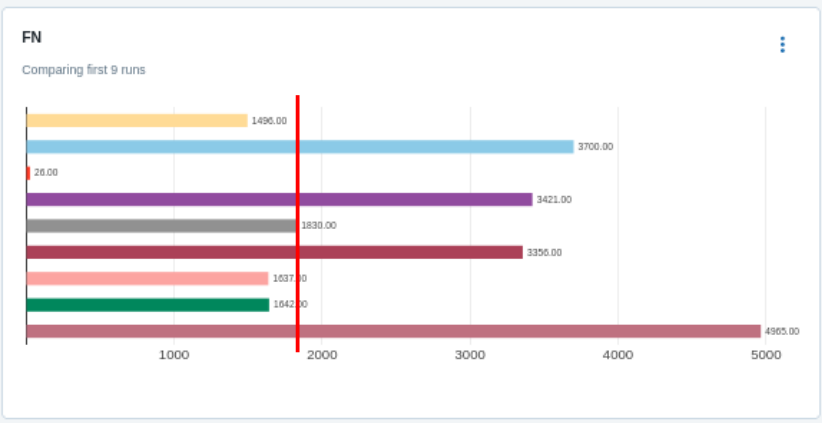
Share

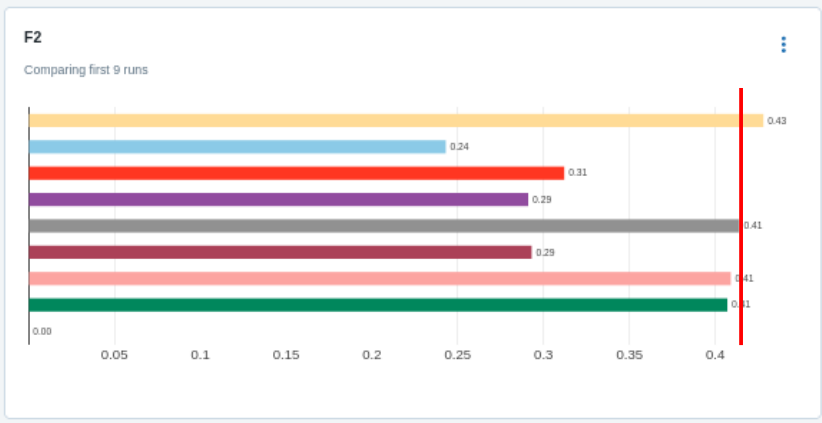
Experiment ID: 373164462905213212 Artifact Location: file:///home/raquelsp/Documents/Openclassrooms/P7_implementez_modele_scoring/P7_travail/P7_scoring_credit/mlruns/373164462905213212

Description Edit







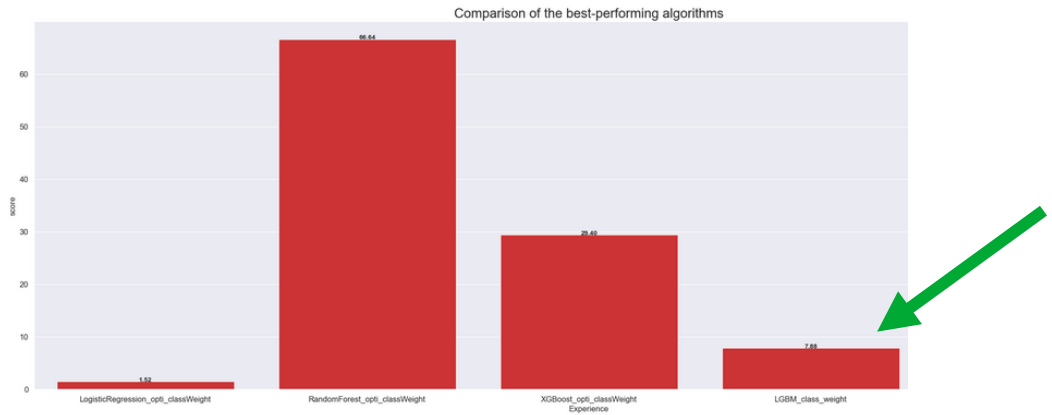


- LGBM_opti_classWeight
- LGBM_opti_smoteUnder
- XGBoost_opti_classWeight
- XGBoost_opti_smoteUnder
- RandomForest_opti_classWeight
- RForestClass_opti_smoteUnder
- LogisticRegression_opti_classWei...
- LogisticRegression_opti_smoteU...
- dummy_class_opti_unbalanced

Comparaison des modèles

Prêt à dépenser

	Run Name	Created	Duration	Metrics AP_SCORE	Accuracy	Business_score	F2	FN	ROC_AUC	Tags others
	LGBM_opti_classWeight	16 days ago	1.6s	0.142	0.696	0.697	0.428	1496	0.697	25 best val...
	LGBM_opti_smoteUnder	16 days ago	1.6s	0.113	0.86	0.699	0.243	3700	0.584	25 best val...
	XGBoost_opti_classWeight	16 days ago	1.5s	0.083	0.114	0.349	0.312	26	0.516	25 best val...
	XGBoost_opti_smoteUnder	16 days ago	1.6s	0.128	0.862	0.715	0.291	3421	0.611	25 best val...
	RandomForest_opti_classWeight	16 days ago	1.6s	0.14	0.729	0.703	0.414	1830	0.685	25 best val...
	RForestClass_opti_smoteUnder	16 days ago	1.4s	0.123	0.848	0.709	0.293	3356	0.609	25 best val...
	LogisticRegression_opti_classWei...	17 days ago	1.5s	0.134	0.689	0.684	0.409	1637	0.68	25 best val...
	LogisticRegression_opti_smoteU...	17 days ago	1.5s	0.133	0.687	0.682	0.407	1642	0.679	25 best val...
	dummy_class_opti_unbalanced	17 days ago	1.7s	0.081	0.919	0.674	0	4965	0.5	25 best val...



Modèle choisi : LightGBM avec class_weight

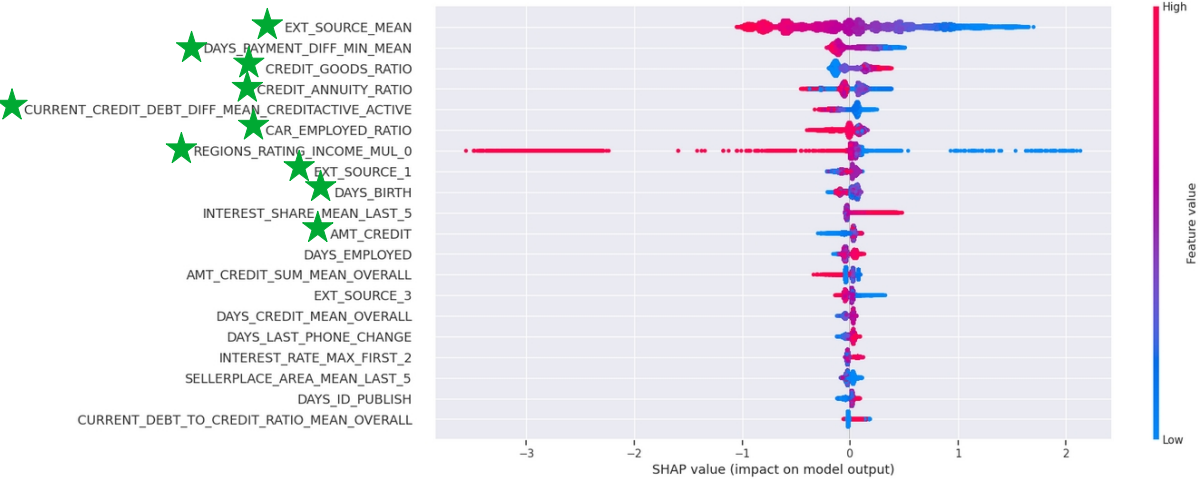
Prêt à dépenser

_best_hyperparamètres : max_depth = 8 (profondeur maximale des arbres construits) ;
min_child_samples = 50 (nombre minimale de clients dans chacune des terminations des arbres)

Seuil de probabilité 0.5

Experience	recall	precision	f2_score	roc_auc	ap_score	business_metric
LogisticRegression_opti_classWeight	0.674000	0.161000	0.411000	0.682000	0.135000	0.685000

Impact des variables sur le modèle, interprétabilité globale

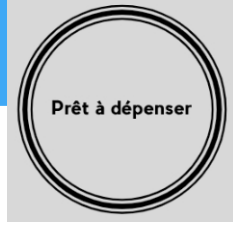


Parmi les variables avec le plus d'impact sur le modèle :

• des **informations bancaires** : *CREDIT_ANNUITY_RATIO* (ratio du montant du crédit du prêt sur l'annuité de prêt font partie des informations), *CREDIT_GOODS_RATIO* (ratio du montant du prêt sur le prix réel du bien), *BUREAU_CURRENT_CREDIT_DEBT_TO_CREDIT_RATIO_MEAN* (le cumul des autres prêts en cours) …,

• les **données externes** : *EXT_SOURCE_MEAN* et *EXT_SOURCE_1*

• les **informations personnelles** : *DAYS_BIRTH*, *CAR_EMPLOYED_RATIO*.



Problématique et données

Modélisation

Pipeline déploiement

Analyse data drift

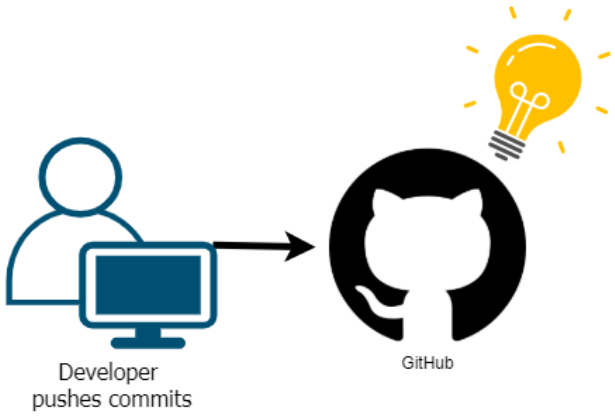
API et Dashboard

Conclusions

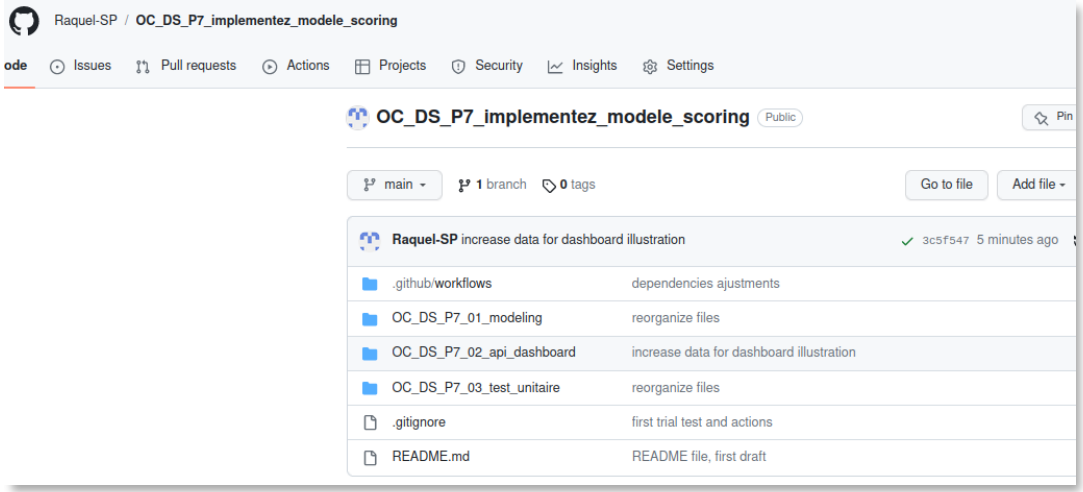
Perspectives

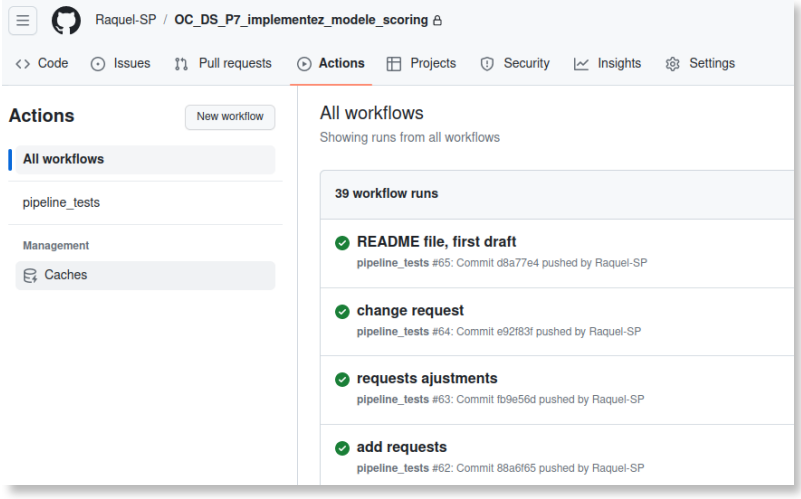
Annexes

Pipeline déploiement



Tests unitaires



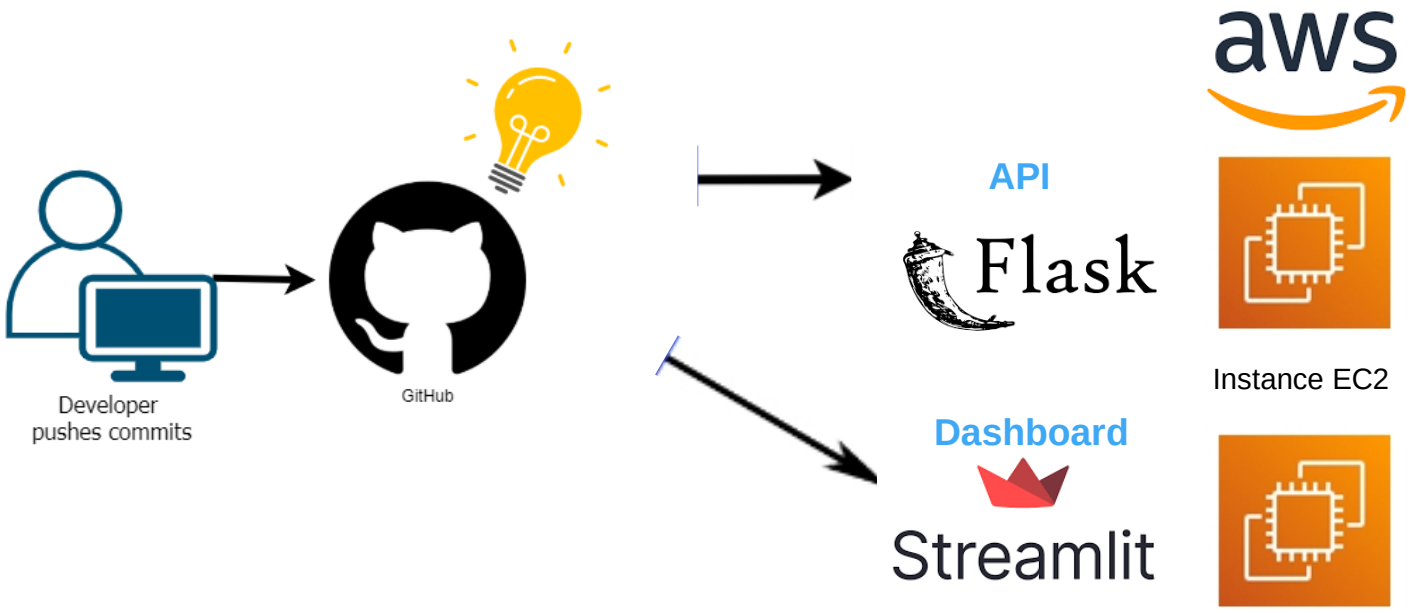


pytest

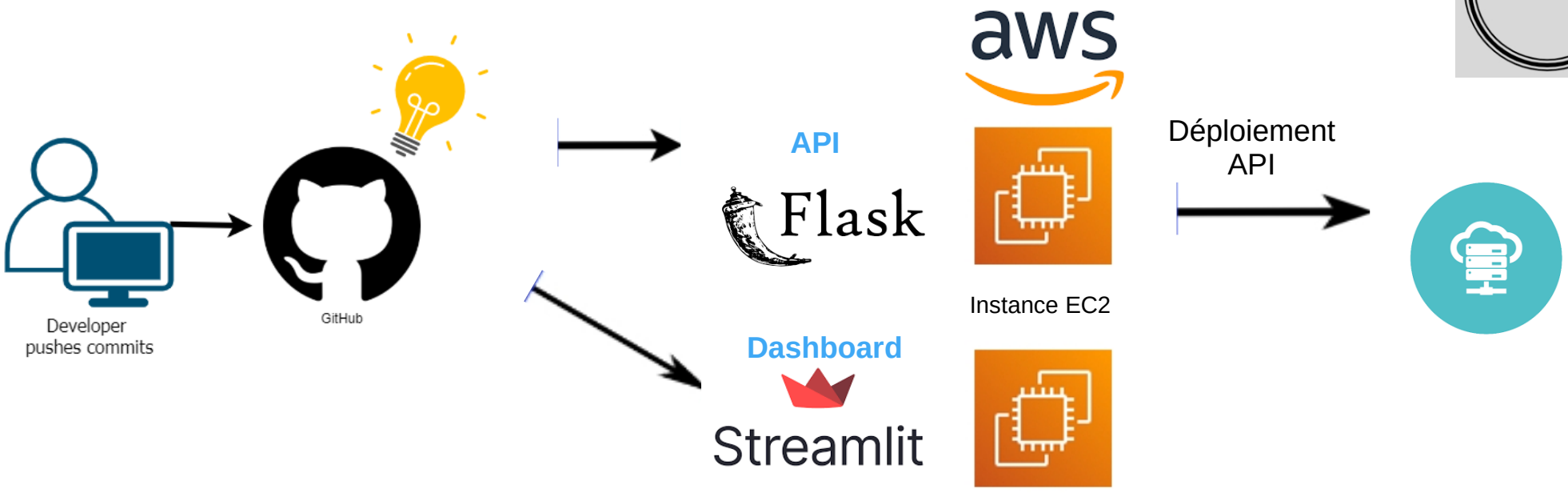
Test unitaires avec **pytest** :

- **Fonction analysant la corrélation** entre les variables et supprimant celles avec une haute corrélation
  - clé dans la sélection des variables
  - clé dans la prévention de l'overfitting
- **Modèle déployé dans l'API**
  - client avec difficultés de paiement > crédit refusé
  - Client sans difficultés de paiement > crédit accordé

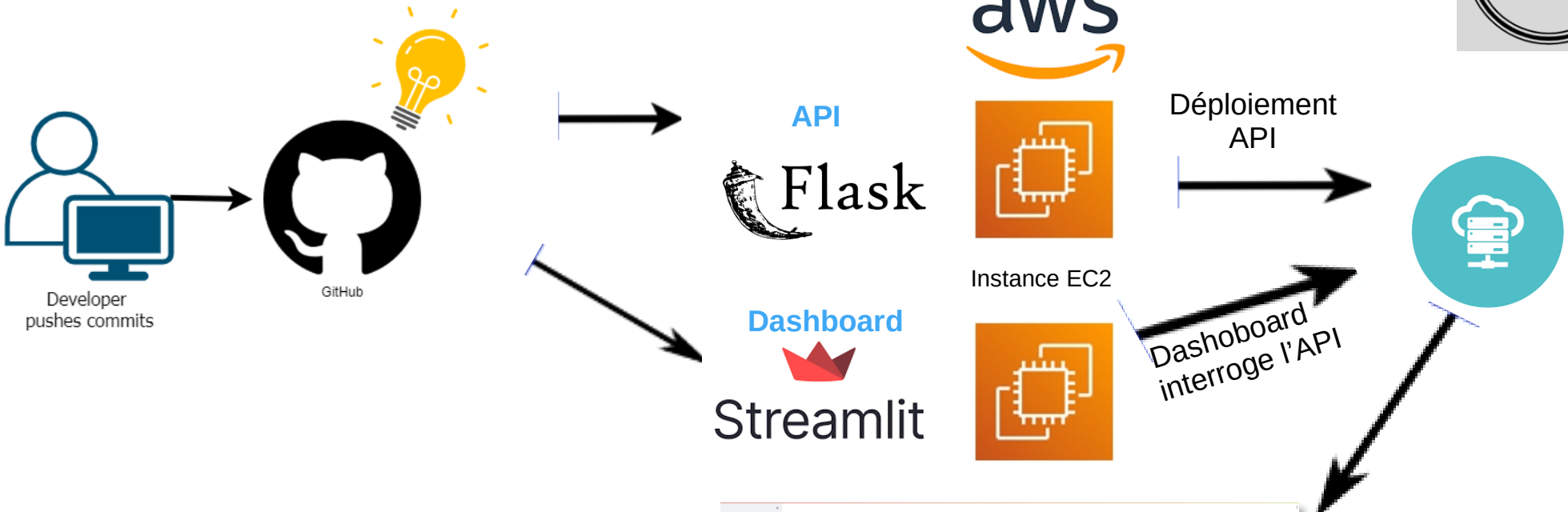
Pipeline déploiement



Pipeline déploiement



Pipeline déploiement



Déploiement du Dashboard

Prêt à dépenser

Problématique et données

Modélisation

Pipeline déploiement

Analyse data drift

API et Dashboard

Conclusions et perspectives

Annexes

Data drift

Prêt à dépenser

Les **données** sur lequel **s'exécute** le modèle peuvent **différer** de façon trop importante des données **d'entraînement**.

Testons les données **application_train vs application_test**

EVIDENTLY AI

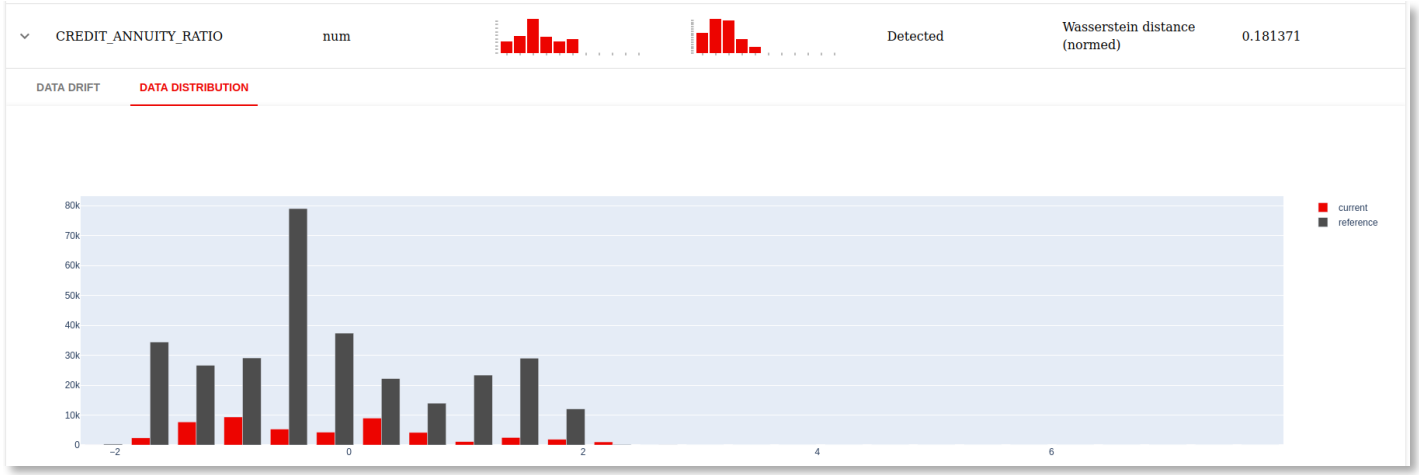
Dataset Drift

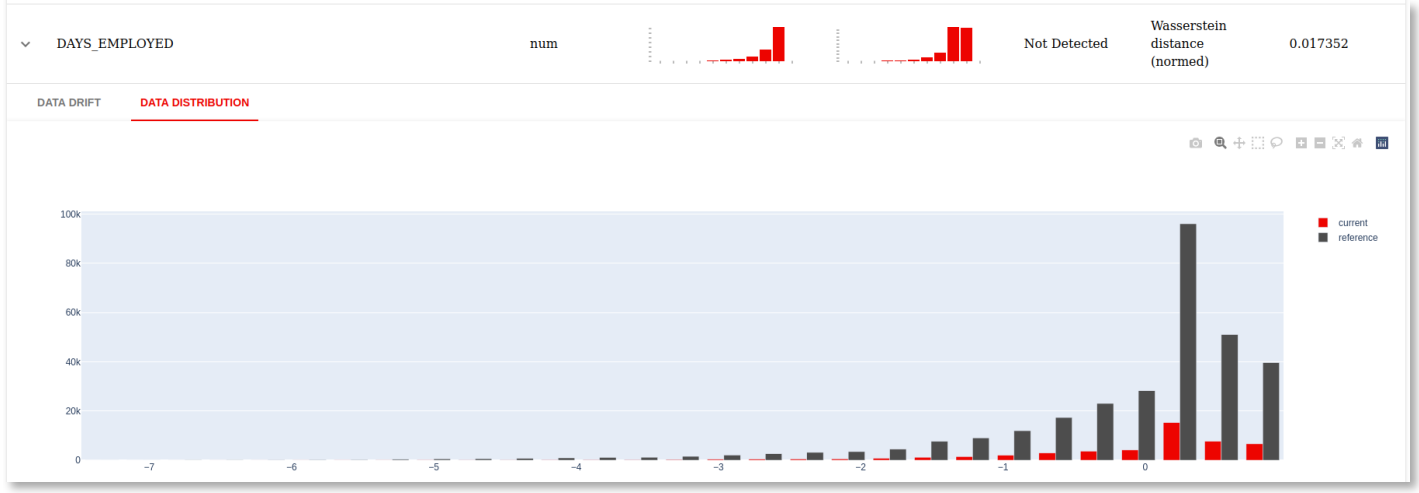
Dataset Drift is NOT detected. Dataset drift detection threshold is 0.5

34	6	0.176
Columns	Drifted Columns	Share of Drifted Columns

Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
CAR_EMPLOYED_RATIO	num			Detected	Wasserstein distance (normed)	0.212644
CREDIT_ANNUITY_RATIO	num			Detected	Wasserstein distance (normed)	0.181371
REGIONS_RATING_INCOME_MUL_0	num			Detected	Wasserstein distance (normed)	0.174787
EXT_SOURCE_1	num			Detected	Wasserstein distance (normed)	0.125958
INTEREST_RATE_MAX_FIRST_2	num			Detected	Wasserstein distance (normed)	0.119802
CREDIT_GOODS_RATIO	num			Detected	Wasserstein distance (normed)	0.103738
AMT_CREDIT	num			Not Detected	Wasserstein distance (normed)	0.09832
SELLERPLACE_AREA_MEAN_LAST_5	num			Not Detected	Wasserstein distance (normed)	0.095042
AMT_CREDIT_SUM_MEAN_OVERALL	num			Not Detected	Wasserstein distance (normed)	0.063588

Data drift





- L'analyse **ne détecte pas la dérive du dataset**.
- Le résultat de l'analyse indique l'existence d'une **dérive dans 6 des 34 variables.**
- **Trois des variables** pour lesquelles la **dérive des données** est détectée, CREDIT_ANNUITY_RATIO, EXT_SOURCE1 et CREDIT_GOODS_RATIO, sont **parmi les 10 variables avec le plus d'importance** dans le modèle.
- Résultats soulignent la **nécessité** d'établir un **protocole** pour la **maintenance et la mise à jour** du modèle.

Prêt à dépenser

Problématique et données

Modélisation

Pipeline déploiement

Analyse data drift

API et Dashboard

Conclusions et perspectives

Annexes

Dashboard Demo

Prêt à dépenser

Prêt à dépenser

Dashboard - Aide à la décision

Comparer avec l'ensemble des clients

Sélectionner les informations à afficher

Choose an option

Distribution des défaillants par variable

☐ Voir les distributions

Prêt à dépenser

Présentation du scoring client et explication du score attribué.

Ce tableau de bord permet de visualiser les informations concernant un client et les comparer avec celles de l'ensemble de la base de données.
Vous devez fournir l'identifiant du client (_SK_ID_CURR_) dans le champ de recherches ci-dessous et vos accèderez aux informations rélatives au score.
La barre à gauche permet l'accées à :

* les comparaisons avec l'ensemble des clients
* la distribution des clients défaillants vs. non défaillants para variable

Selectionner l'identifiant du client

Selectionner l'identifiant du client

177502

Information sur le client

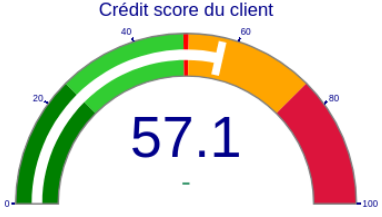
	AGE	GENDER	NAME_FAMILY_STATUS	CNT_CHILDREN	NAME_EDUCATION_TYPE	NAME_INCOME_TYPE	YEARS_WORKING
177502	32	Man	Married	0	Secondary / secondary special	Working	0

	AMT_INCOME_TOTAL	NAME_CONTRACT_TYPE	AMT_GOODS_PRICE	NAME_HOUSING_TYPE
177502	135,000.0000	Cash loans	900,000.0000	House / apartment

Score du client

Crédit refusé

Crédit score du client



Variables ayant le plus d'impact sur le score du client

Ouvrir pour afficher l'analyse détaillée des variables avec le plus d'influence

Prêt à dépenser

Problématique et données

Modélisation

Pipeline déploiement

Analyse data drift

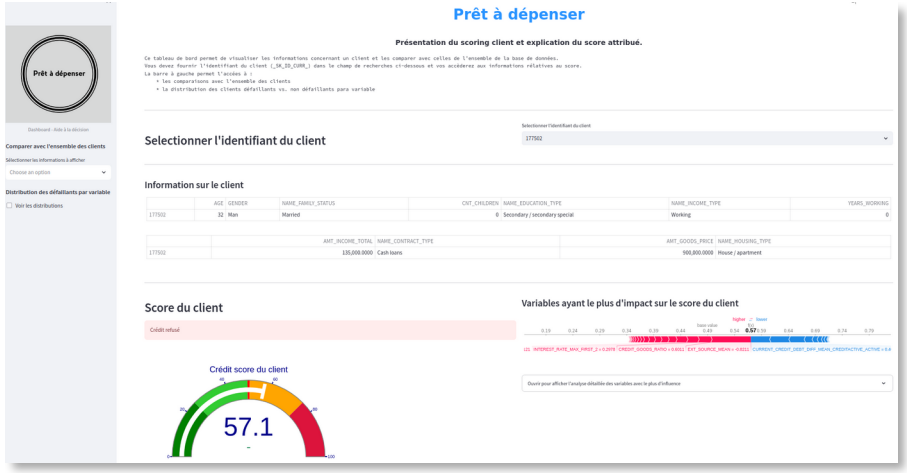
API et Dashboard

Conclusions et perspectives

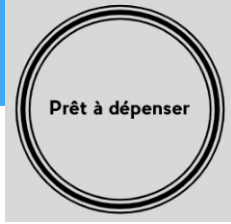
Annexes

Conclusions

- Proposition d’un modèle **classification binaire avec classes déséquilibrées**
  - On prends en compte le besoin **minimiser** le nombre de **FN** (faux négatifs, clients classés comme non défaillants, étant défaillants), création et optimisation d’un **indice métier**
  - Modèle final **LightGBM**, avec **class_weight** et **hyperparamètres optimisés** (max_depth = 8, min_child_samples = 50)
  - **Dérive de données** 3 des 10 variables avec le plus d’impact sur le modèle, importance de veiller à la **mise à jour**
- Une **API et un dashboard** ont été **déployés** pour faciliter la **communication avec les clients**



Perspectives



- Des **informations plus détaillées** sur le montant **gagné** lorsque le prêt est **remboursé**, et montant **perdu** lorsque le prêt est **non remboursé**, seraient extrêmement utiles pour **optimiser le indice métier développé.**
- De toute évidence on souhaiterait avoir **plus d'échantillons de clients défaillants** pour résoudre le problème du déséquilibre dans la cible.
- Une **collaboration** avec notre **client** permettrait d'enrichir notre modèle de leur expérience.
  - Les experts métiers pourraient nous donner leur avis sur l'intérêt des nouvelles variables créées et pourquoi pas nous orienter vers des nouvelles variables.
  - Une **explication des données externe** serait un plus puisqu'il est difficile d'être transparent en utilisant ces variables importantes pour le modèle mais inexplicables pour le client.

Prêt à dépenser

Problématique et données

Modélisation

Pipeline déploiement

Analyse data drift

API et Dashboard

Conclusions et perspectives

Annexes

Annexe : Processus modélisation

Prêt à dépenser

Préparation des données

Données originaux

Type de données
Valeurs aberrantes
Valeurs manquantes
Encodage
Merge
Feature engineering
Feature selection

Données pour entraîner le modèle

Sélection des métriques

Sélection à partir de littérature

- Recall
- Précision
- F-beta
- ROC AUC
- Average Precision

Développement indice métier

Prendre en compte le différent impact économique des différents erreurs de classification

Tests algorithmes

Rééquilibrage de la variable cible

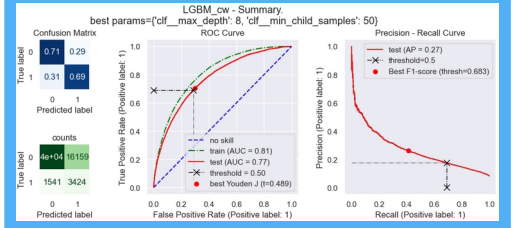
- Smote
- Smote + RandomUnderSampler
- Class weigth

Optimisation hyperparamètres

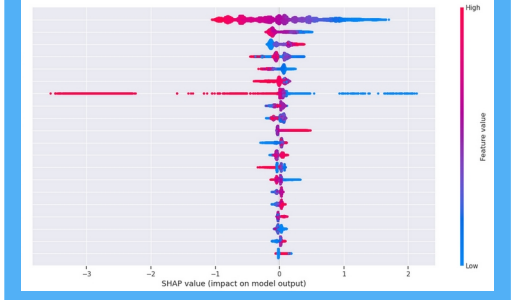
- Cinq algorithmes
- Grid Search

Meilleur modèle

Hyperparamètres et performance



Interprétabilité

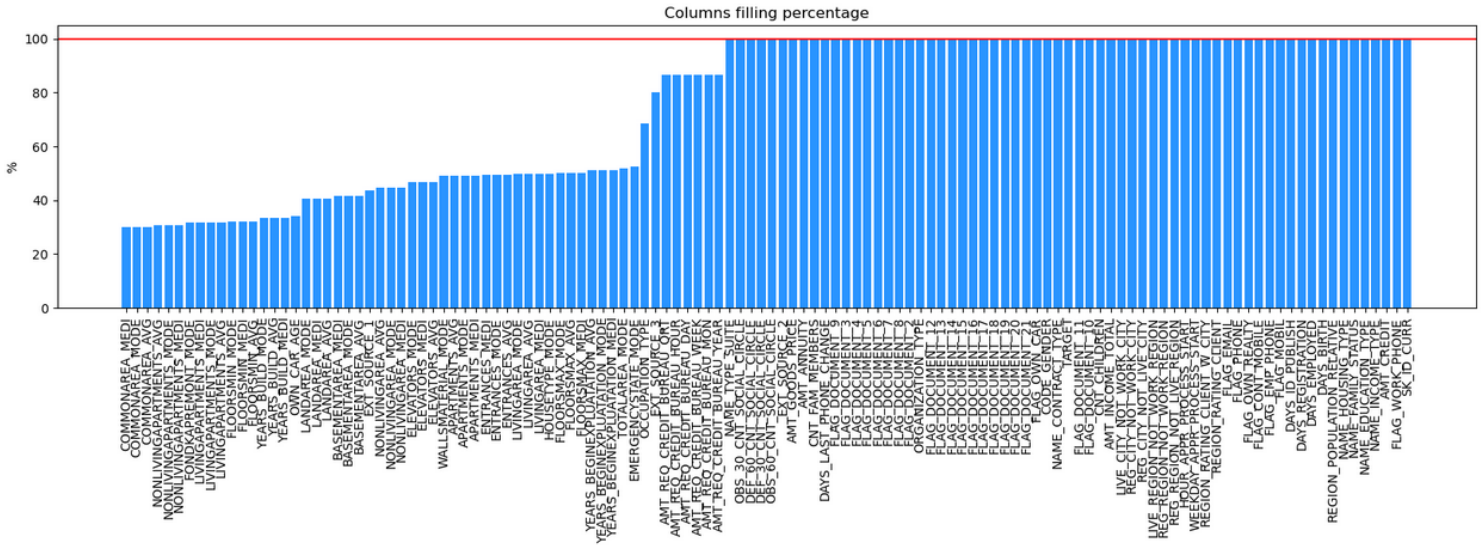


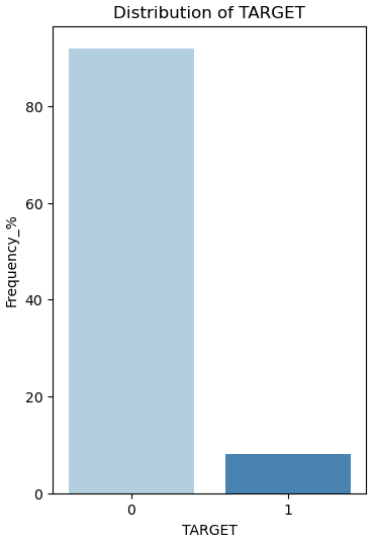
Données disponibles

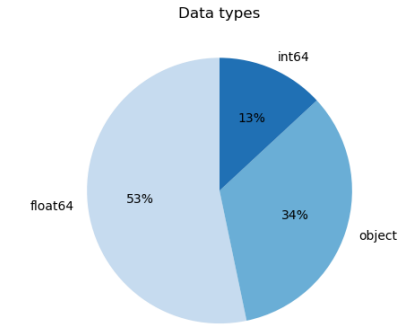
Prêt à dépenser

Exemple exploration application_train

Target	Signification	Objectif	Signification
1	Client avec difficultés de paiement	1	Crédit refusé
0	Autres cas	0	Crédit accordé





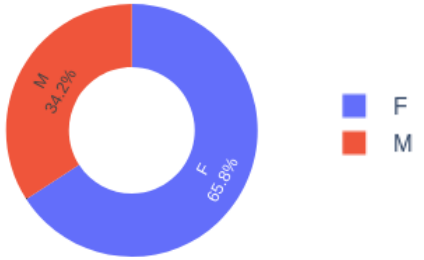


Annexe : Exemple exploration application_train

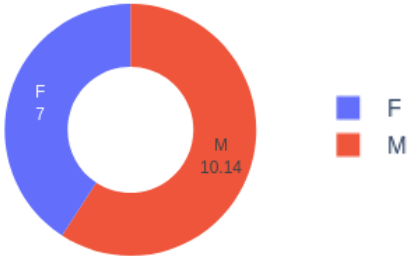
Variable qualitative

```
------------------------------------------------
The unique categories of 'CODE_GENDER' are:
['M' 'F' 'XNA']
------------------------------------------------
Counts of each category are:
F      202448
M      105059
XNA         4
Name: CODE_GENDER, dtype: int64
------------------------------------------------
```

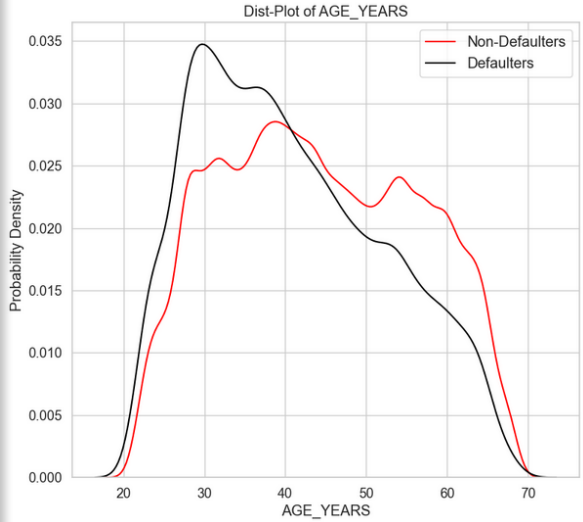
Distribution of CODE_GENDER for all Targets

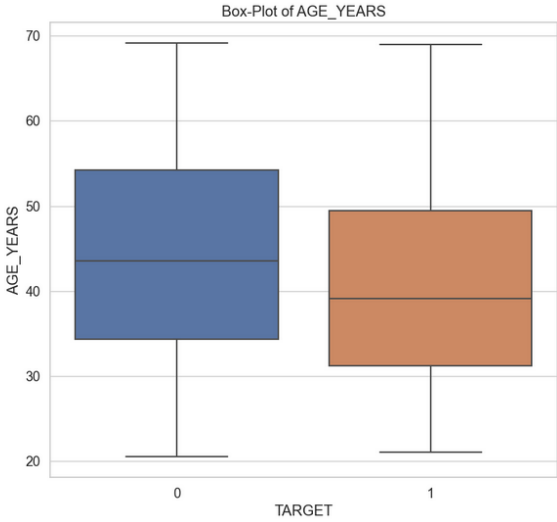


Percentage of Defaulters for each category of CODE_GENDER



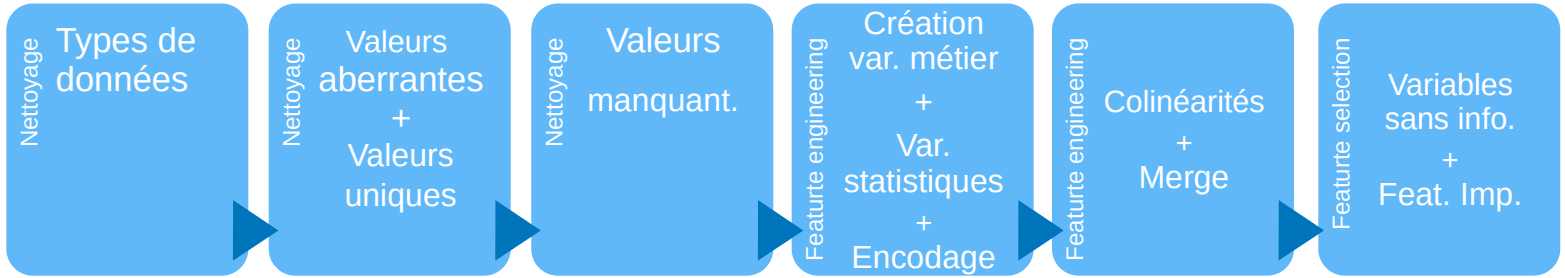
Variable quantitative





Préparation des données

Prêt à dépenser



- Objets Y/N ou Yes/No en int8
- Numérique ordinal en objet
- Réduction pour optimiser la mémoire

- 9 fichiers
- Variables différents types
- Valeurs manquants
- Valeurs aberrantes
- ...

- Correction des valeurs aberrantes

- Suppression si nan >68 %
- Imputation quanti : médiane ou 0 ou NaNImputer
- Imputation quali : mode ou XNA

- 9 fichiers
- Variables différents types
- ~~Valeurs manquants~~
- ~~Valeurs aberrantes~~
- ...

- Métier : création manuelle
- Statistiques : min, max…
- Encodage quanti. : StandardScaler
- Encodage quali. : LabelEncoder et get_dummies
- Détails en annexe

- Suppr. Variables correl > 0.7 (coef. Pearson)

- 2 fichiers (train, test)
- Pas de nan
- 417 variables
- Float & int

- Suppr. Variables 1 seul valeur
- Feat importance LightGBM + RandomForest

2 fichiers (train, test)

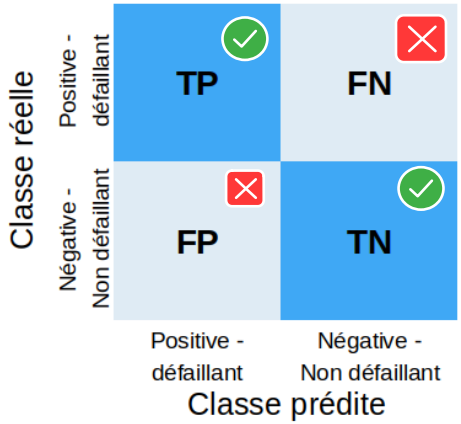
Pas de variables sans information

34 variables, les 25 variables avec le plus d'importance LightGBM & Random Forest

Annexe : Feature engineering

Automatique, création de variables statistiques	• Variables quantitatives : min, max, sum, mean, var • Variables qualitatives : Sum, count, mean
Manuel	• Revenu de rente et de crédit : ratio / différence • Jours en années, changement de jours : ratio • Âge de la voiture, ancienneté d'emploi : ratio / différence • Flag sur les téléphones : ratio / différence • Membres de la familie : ratio / différence • Note dé la région de résidence : ratio / différence • Données externes : ratio, moyenne, max, min • Informations sur le bâtiment : somme, multiplication • Défauts de paiements et défauts observables : somme / ratio • Flag sur les documents : somme, moyenne, variance, écart-type • Modification du demandeur : somme /ratio

Annexe : Sélection des métriques



Metrics

- **Recall**: the metric for determining the **true positive rate** (TP), which measures how many of all the positive observations have been classified as positive. To avoid losses, we need to detect all defaulters (positive class) and therefore **maximise the recall** metric.

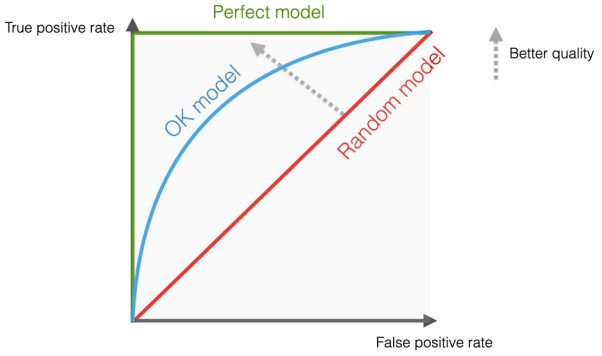
$$RecallScore = \frac{TP}{FN + TP}$$

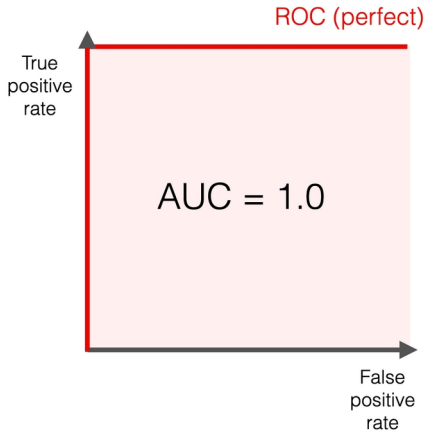
- **Precision**: this measures the number of observations predicted as positive (defaulting customer) that are actually positive. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative. If a customer is predicted to default when in fact they do not, the loan will not be granted and interest will not be paid. You therefore need to **maximise 'Precision'**.

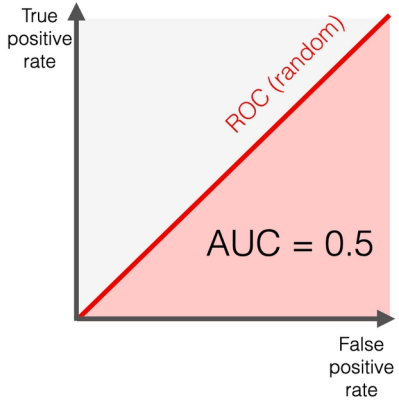
$$PrecisionScore = \frac{TP}{FP + TP}$$

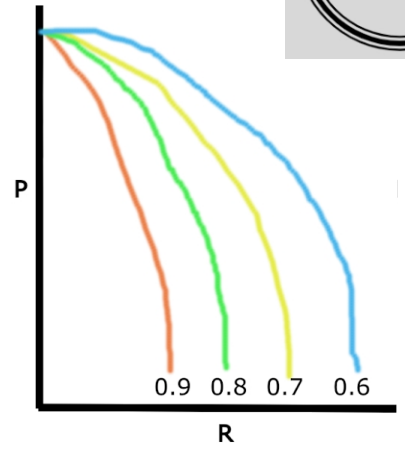
- **F-measure or F1**: a harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal.
  - In our case, we need to find the greatest number of truly positive observations (customer predicted as failing and actually failing) and the loss is less if a failing customer is predicted but does not actually fail (false positives), so **we will give priority to maximising recall at the expense of precision** (we are talking about Precision, not accuracy).
  - Setting the beta parameter for the Fbeta score gives more weight to recall (beta>1) than to precision (0<beta<1).

Annexe : Sélection des métriques







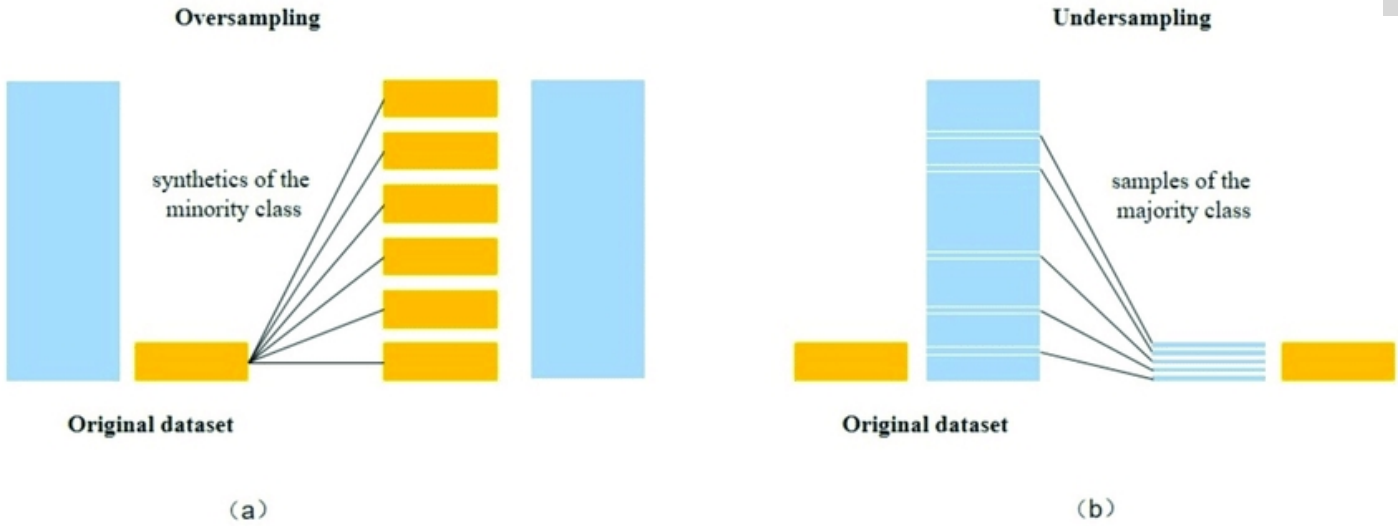


IoU Threshold Set : [0.6, 0.7, 0.8, 0.9]

Metrics

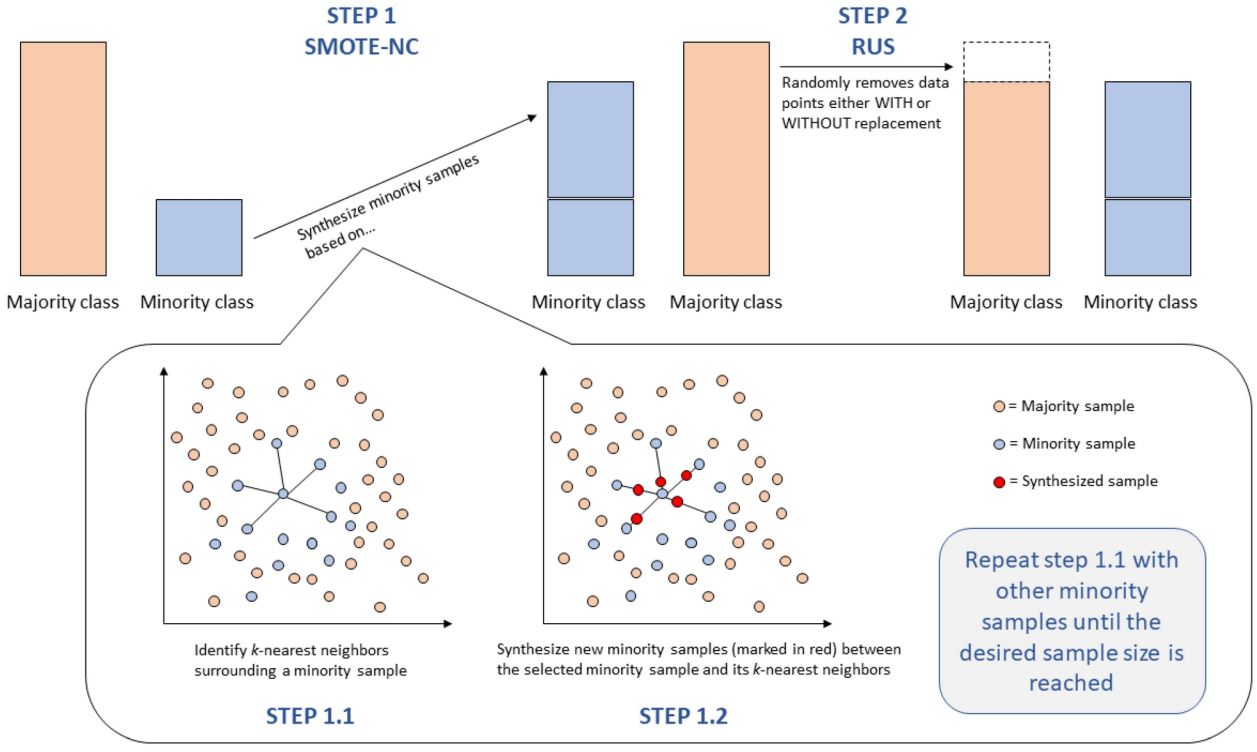
- **ROC AUC score**: the ROC AUC (Area Under the Receiver Operating Characteristic Curve) score is equivalent to calculating the rank correlation between the predictions and the target. From an interpretation point of view, it is more useful because it tells us that this metric shows how good your model is at ranking predictions. It tells you the probability that a randomly selected positive instance will be ranked higher than a randomly selected negative instance.
- **Score AP** : Compute average precision (AP) from prediction scores. AP summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight. A utiliser :
  - lorsque vous voulez communiquer la décision de précision/rappel à d'autres parties prenantes et que vous voulez choisir le seuil qui correspond au problème de l'entreprise.
  - lorsque vos données sont fortement déséquilibrées. Puisque l'AUC de PR se concentre principalement sur la classe positive (PPV et TPR), elle se soucie moins de la classe négative fréquente.
  - when you care more about the positive class than the negative class. If you care more about the positive class and therefore the PPV and TPR, you should opt for the Precision-Recall curve and the PR AUC (average precision).

Annexe : Class imbalance



Wei Xia et al. 2019 High-Resolution Remote Sensing Imagery Classification of Imbalanced Data Using Multistage Sampling Method and Deep Neural Networks

Annexe : Class imbalance



Hybrid resampling process

Tarid Wongvorachan et al. 2023 A Comparison of Undersampling, Oversampling, and SMOTE Methods for Dealing with Imbalanced Classification in Educational Data Mining

Annexes : Algorithmes considérés

Dummy (Baseline)	most frequent

DummyClassifier makes predictions that ignore the input features.

This classifier serves as a simple baseline to compare against other more complex classifiers.

The specific behavior of the baseline is selected with the `strategy` parameter.

All strategies make predictions that ignore the input feature values passed as the `X` argument to `fit` and `predict`. The predictions, however, typically depend on values observed in the `y` parameter passed to `fit`.

Note that the “stratified” and “uniform” strategies lead to non-deterministic predictions that can be rendered deterministic by setting the `random_state` parameter if needed. The other strategies are naturally deterministic and, once fit, always return the same constant prediction for any value of `X`.

Read more in the User Guide.

New in version 0.13.

Parameters: **strategy : *{“most_frequent”, “prior”, “stratified”, “uniform”, “constant”}, default=”prior”***

Strategy to use to generate predictions.

- “most_frequent”: the `predict` method always returns the most frequent class label in the observed `y` argument passed to `fit`. The `predict_proba` method returns the matching one-hot encoded vector.

From https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html

Annexes : Algorithmes considérés

Prêt à dépenser

Logistic Regression	Penalty : L1, L2, elasticnet Solver : liblinear

Logistic Regression (aka logit, MaxEnt) classifier.

In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to 'ovr', and uses the cross-entropy loss if the 'multi_class' option is set to 'multinomial'. (Currently the 'multinomial' option is supported only by the 'lbfgs', 'sag', 'saga' and 'newton-cg' solvers.)

This class implements regularized logistic regression using the 'liblinear' library, 'newton-cg', 'sag', 'saga' and 'lbfgs' solvers. **Note that regularization is applied by default**. It can handle both dense and sparse input. Use C-ordered arrays or CSR matrices containing 64-bit floats for optimal performance; any other input format will be converted (and copied).

The 'newton-cg', 'sag', and 'lbfgs' solvers support only L2 regularization with primal formulation, or no regularization. The 'liblinear' solver supports both L1 and L2 regularization, with a dual formulation only for the L2 penalty. The Elastic-Net regularization is only supported by the 'saga' solver.

Read more in the User Guide.

Parameters:

penalty : *{'l1', 'l2', 'elasticnet', None}, default='l2'*

Specify the norm of the penalty:

- `None`: no penalty is added;
- `'l2'`: add a L2 penalty term and it is the default choice;
- `'l1'`: add a L1 penalty term;
- `'elasticnet'`: both L1 and L2 penalty terms are added.

> **Warning:** Some penalties may not work with some solvers. See the parameter `solver` below, to know the compatibility between the penalty and solver.

solver : *{'lbfgs', 'liblinear', 'newton-cg', 'newton-cholesky', 'sag', 'saga'}, default='lbfgs'*

Algorithm to use in the optimization problem. Default is 'lbfgs'. To choose a solver, you might want to consider the following aspects:

- For small datasets, 'liblinear' is a good choice, whereas 'sag' and 'saga' are faster for large ones;
- For multiclass problems, only 'newton-cg', 'sag', 'saga' and 'lbfgs' handle multinomial loss;
- 'liblinear' is limited to one-versus-rest schemes.
- 'newton-cholesky' is a good choice for `n_samples` >> `n_features`, especially with one-hot encoded categorical features with rare categories. Note that it is limited to binary classification and the one-versus-rest reduction for multiclass classification. Be aware that the memory usage of this solver has a quadratic dependency on `n_features` because it explicitly computes the Hessian matrix.

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Annexes : Algorithmes considérés

Prêt à dépenser

Random forest	max_depth : 5, 10, 50, None max_features : auto, sqrt Criterion : gini, entropy

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Parameters:

criterion : *{"gini", "entropy", "log_loss"}, default="gini"*

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain, see Mathematical formulation. Note: This parameter is tree-specific.

max_depth : *int, default=None*

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

max_features : *{"sqrt", "log2", None}, int or float, default="sqrt"*

The number of features to consider when looking for the best split:

- If int, then consider `max_features` features at each split.
- If float, then `max_features` is a fraction and `max(1, int(max_features * n_features_in_))` features are considered at each split.
- If "sqrt", then `max_features=sqrt(n_features)`.
- If "log2", then `max_features=log2(n_features)`.
- If None, then `max_features=n_features`.

Changed in version 1.1: The default of `max_features` changed from `"auto"` to `"sqrt"`.

Note: the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than `max_features` features.

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Annexes : Algorithmes considérés

XGBoost	learning_rate : 0.1, 0.01, 0.05 Gamma : 0, 0.5, 2, 5, Subsample : 0.6, 1.0 max_depth : 4, 6

XGBoost Documentation

XGBoost is an optimized distributed gradient boosting library designed to be highly **efficient**, **flexible** and **portable**. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.

Parameters for Tree Booster

- `eta` [default=0.3, alias: `learning_rate`]
  - Step size shrinkage used in update to prevents overfitting. After each boosting step, we can directly get the weights of new features, and `eta` shrinks the feature weights to make the boosting process more conservative.
  - range: [0,1]
- `gamma` [default=0, alias: `min_split_loss`]
  - Minimum loss reduction required to make a further partition on a leaf node of the tree. The larger `gamma` is, the more conservative the algorithm will be.
  - range: [0,∞]
- `max_depth` [default=6]
  - Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit. 0 indicates no limit on depth. Beware that XGBoost aggressively consumes memory when training a deep tree. `exact` tree method requires non-zero value.
  - range: [0,∞]
- `subsample` [default=1]
  - Subsample ratio of the training instances. Setting it to 0.5 means that XGBoost would randomly sample half of the training data prior to growing trees. and this will prevent overfitting. Subsampling will occur once in every boosting iteration.
  - range: (0,1]

https://xgboost.readthedocs.io/en/stable/parameter.html

Annexes : Algorithmes considérés

LightGBM	max_depth : 10, 50, 150 n_estimators : 100, 200 learning_rate : 0.01, 0.01, 1

Construct a gradient boosting model.

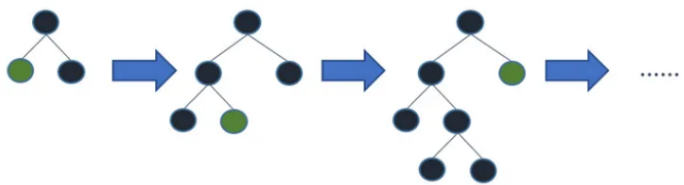
Parameters:

- **max_depth** (*int, optional (default=-1)*) – Maximum tree depth for base learners, <=0 means no limit.
- **learning_rate** (*float, optional (default=0.1)*) – Boosting learning rate. You can use `callbacks` parameter of `fit` method to shrink/adapt learning rate in training using `reset_parameter` callback. Note, that this will ignore the `learning_rate` argument in training.
- **n_estimators** (*int, optional (default=100)*) – Number of boosted trees to fit.

https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html

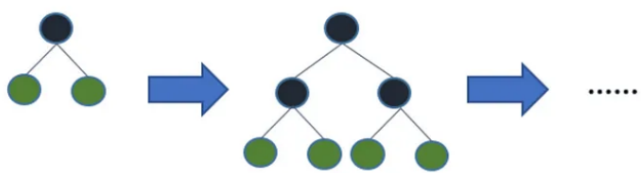
Light GBM grows tree vertically while other algorithm grows trees horizontally meaning that Light GBM grows tree **leaf-wise** while other algorithm grows level-wise. It will choose the leaf with max delta loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm.

Below diagrams explain the implementation of LightGBM and other boosting algorithms.



Leaf-wise tree growth

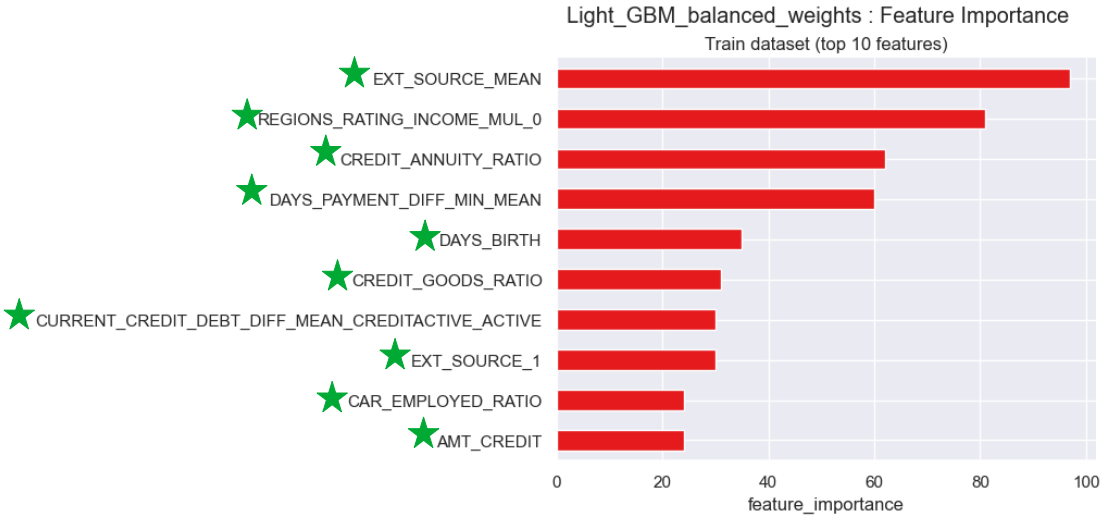
Explains how LightGBM works

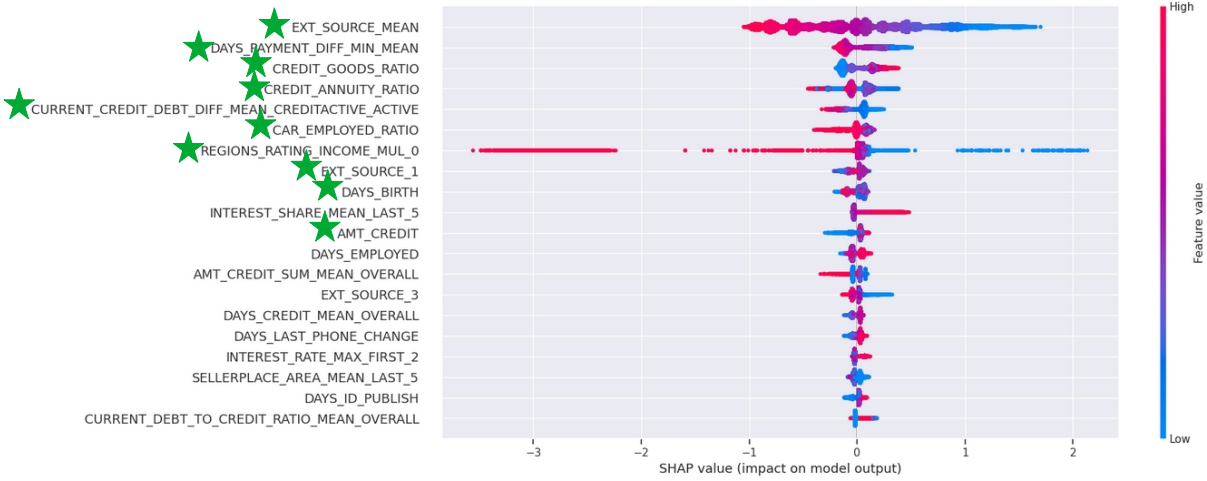


Level-wise tree growth

https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc

Interprétabilité globale





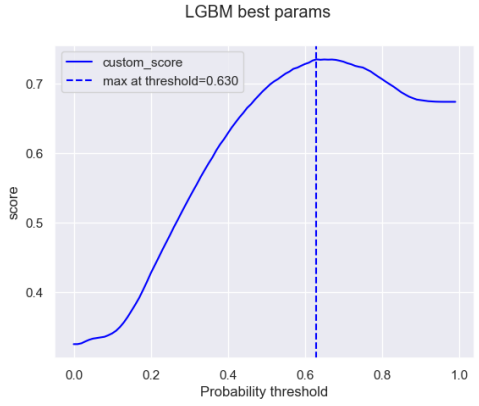
Parmi les variables avec le plus d'impact sur le modèle :

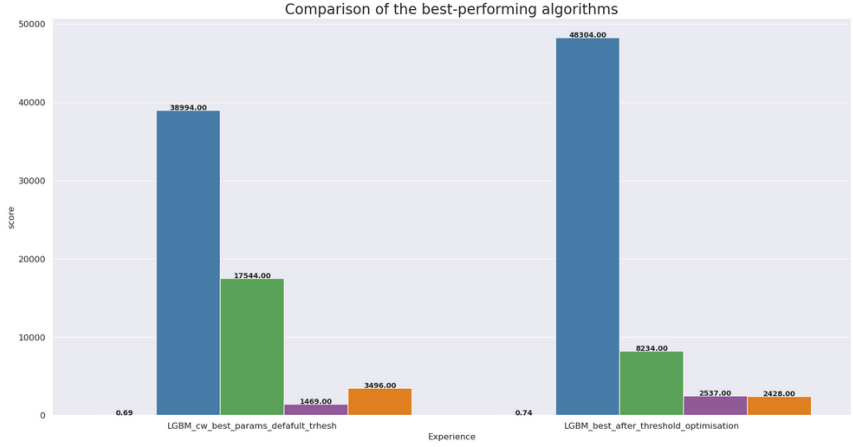
- des **informations bancaires** : *CREDIT_ANNUITY_RATIO* (ratio du montant du crédit du prêt sur l'annuité de prêt font partie des informations), *CREDIT_GOODS_RATIO* (ratio du montant du prêt sur le prix réel du bien), *BUREAU_CURRENT_CREDIT_DEBT_TO_CREDIT_RATIO_MEAN* (le cumul des autres prêts en cours) …,
- les **données externes** : *EXT_SOURCE_MEAN* et *EXT_SOURCE_1*
- les **informations personnelles** : *DAYS_BIRTH*, *CAR_EMPLOYED_RATIO*.

Annexe : Ajustement seuil probabilité

LightGBM avec class_weight

Ajustement du seuil probabilité pour optimiser le métrique métier





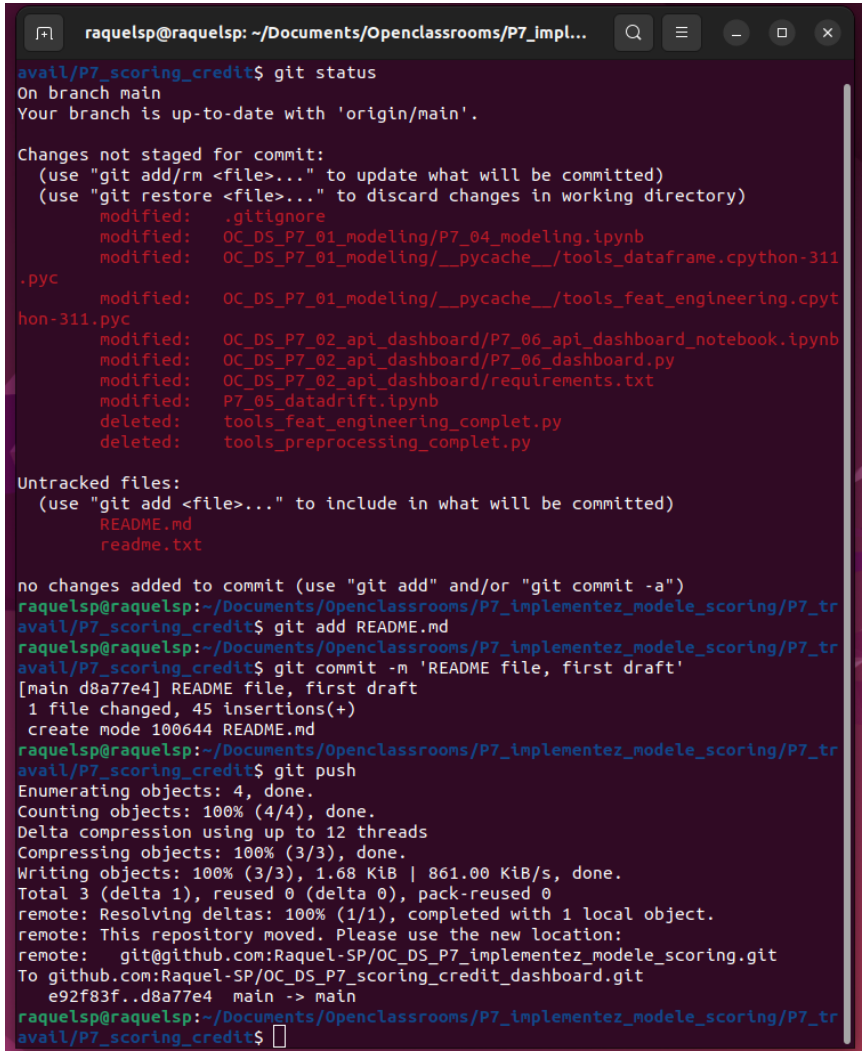
Experience	Business_score	TN	FP	FN	TP
LGBM_cw_best_params_defafult_trhesh	0.694396	38994	17544	1469	3496
LGBM_best_after_threshold_optimisation	0.735376	48304	8234	2537	2428

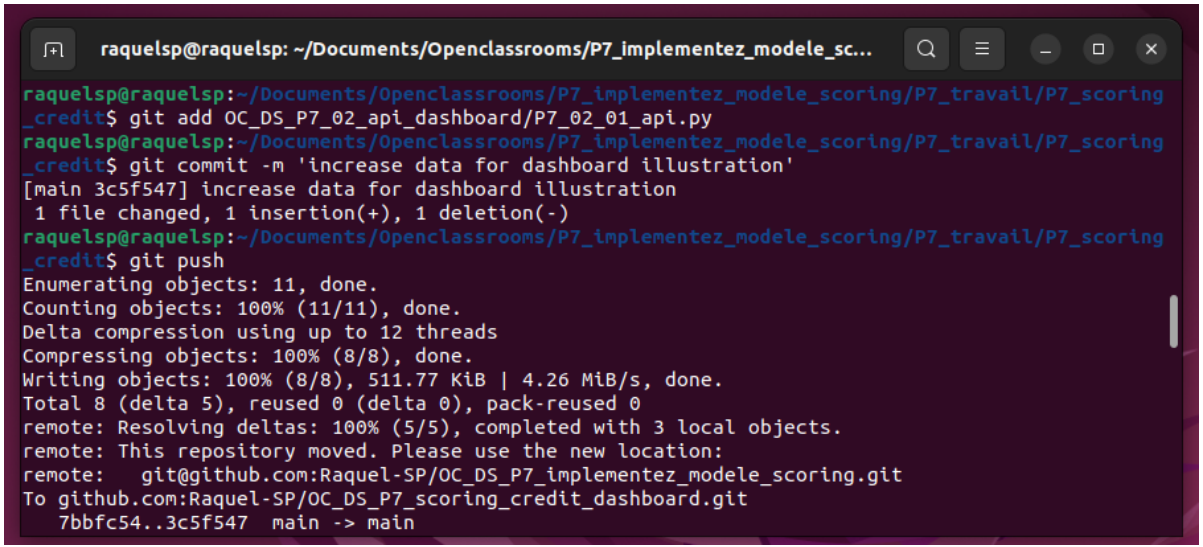
_best_hyperparamètres : max_depth = 8 (profondeur maximale des arbres construits) ;
min_child_samples = 50 (nombre minimale de clients dans chacune des terminations des arbres)

Seuil de probabilité 0.5

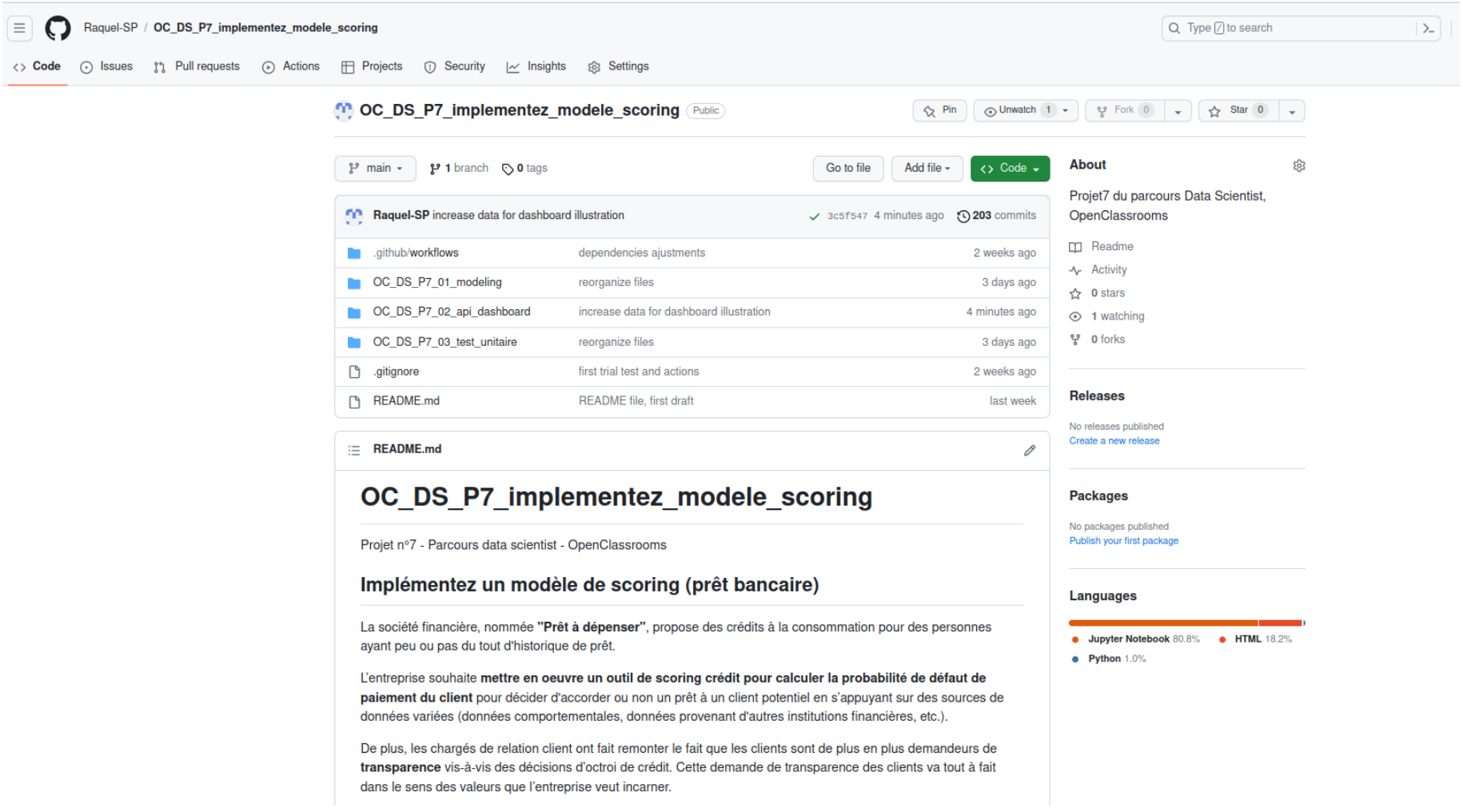
Experience	recall	precision	f2_score	roc_auc	ap_score	business_metric
LogisticRegression_opti_classWeight	0.674000	0.161000	0.411000	0.682000	0.135000	0.685000

Copie d’écran des commits vers Github



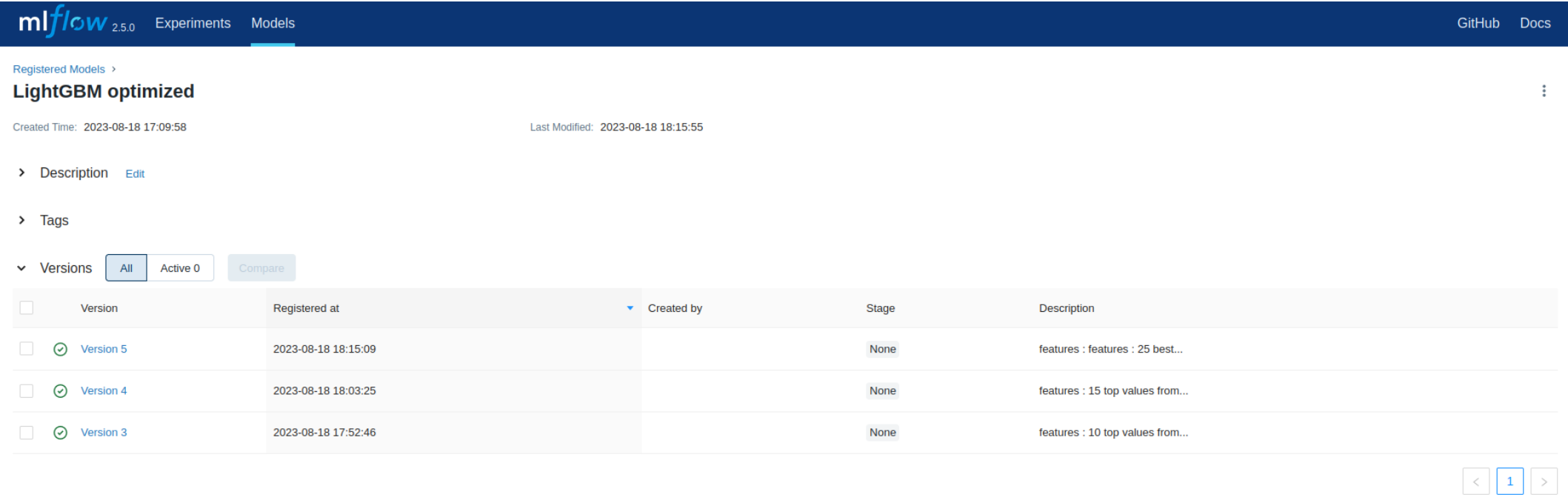


Copie d'écran du dossier Github

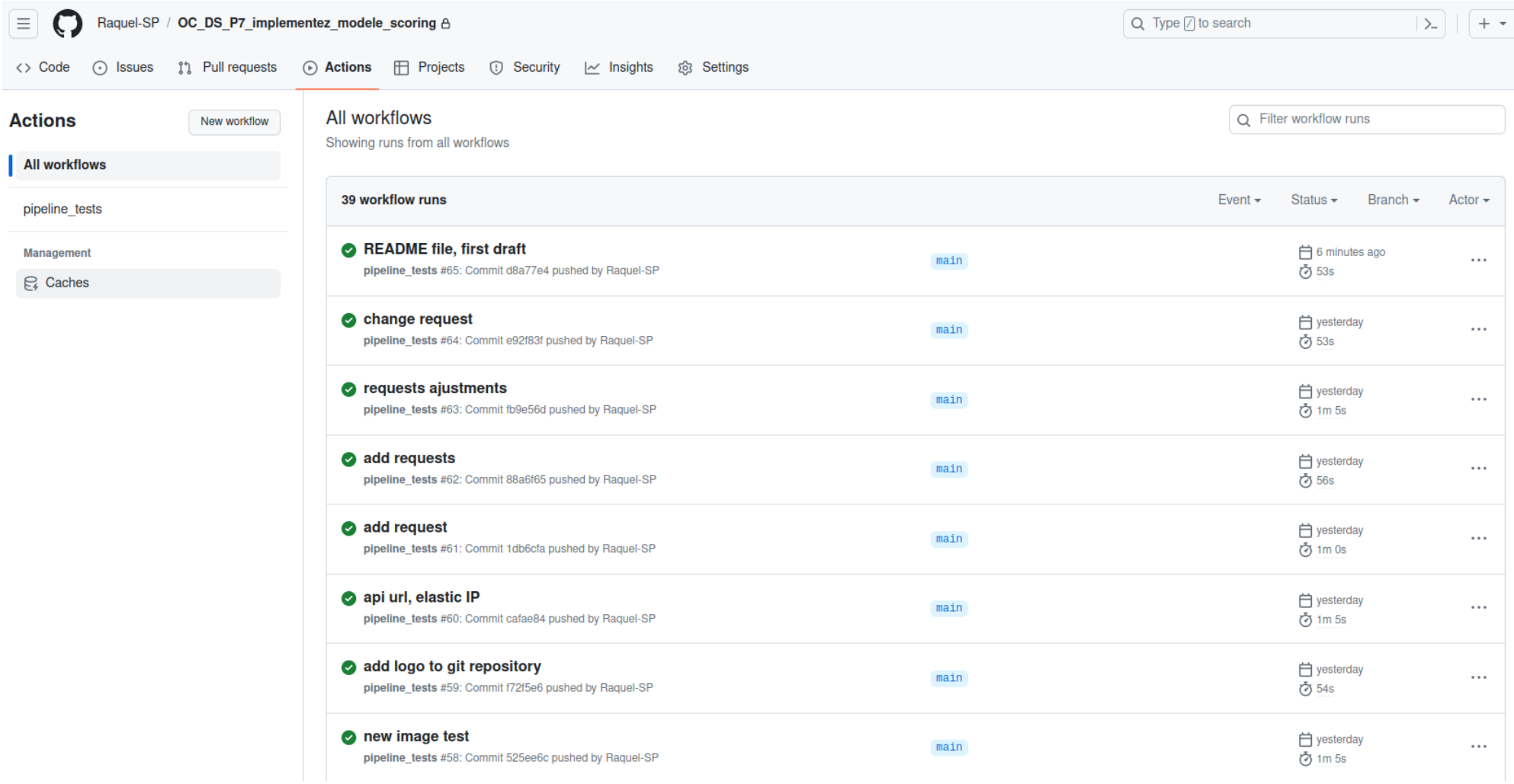


https://github.com/Raquel-SP/OC_DS_P7_implementez_modele_scoring

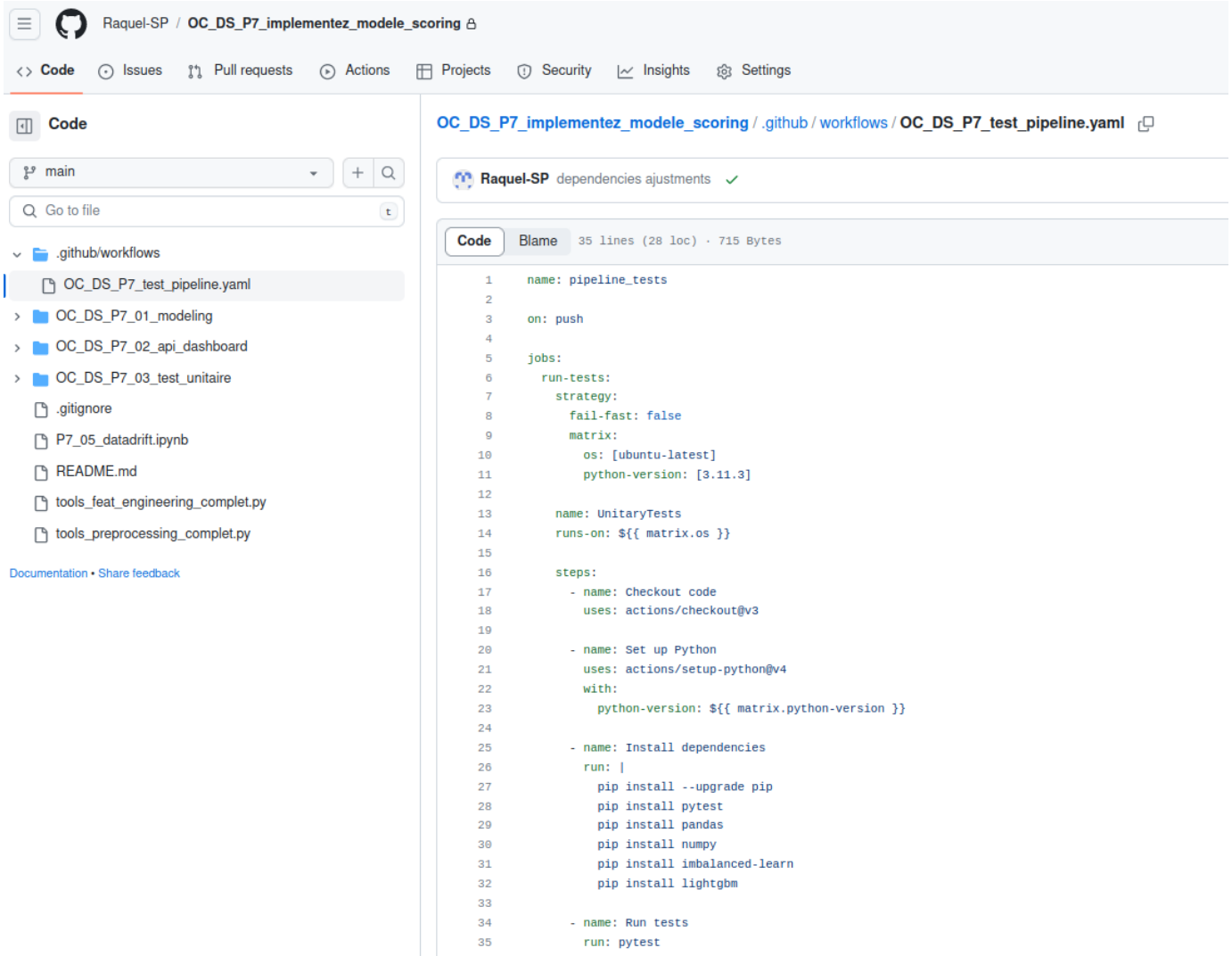
Copie d’écran versions du modèle



Copie d’écran des test unitaires



Copie d’écran des test unitaires



Raquel-SP / **OC_DS_P7_implementez_modele_scoring**

Code | Issues | Pull requests | Actions | Projects | Security | Insights | Settings

Code

main

Go to file

- .github/workflows
  - OC_DS_P7_test_pipeline.yaml
- OC_DS_P7_01_modeling
- OC_DS_P7_02_api_dashboard
- OC_DS_P7_03_test_unitaire
- .gitignore
- P7_05_datadrift.ipynb
- README.md
- tools_feat_engineering_complet.py
- tools_preprocessing_complet.py

Documentation • Share feedback

OC_DS_P7_implementez_modele_scoring / .github / workflows / **OC_DS_P7_test_pipeline.yaml**

Raquel-SP dependencies ajustments ✓

Code | Blame | 35 lines (28 loc) · 715 Bytes

```yaml
name: pipeline_tests

on: push

jobs:
  run-tests:
    strategy:
      fail-fast: false
      matrix:
        os: [ubuntu-latest]
        python-version: [3.11.3]

    name: UnitaryTests
    runs-on: ${{ matrix.os }}

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Set up Python
        uses: actions/setup-python@v4
        with:
          python-version: ${{ matrix.python-version }}

      - name: Install dependencies
        run: |
          pip install --upgrade pip
          pip install pytest
          pip install pandas
          pip install numpy
          pip install imbalanced-learn
          pip install lightgbm

      - name: Run tests
        run: pytest
```

Annexe : Pipeline déploiement continu

Création (ou modification) du code

Tracking avec **MLflow**

« ***git push*** » dépôt GitHub à jour

Test unitaire, via **GitHub actions** (à chaque *push*)

« ***git pull*** » depuis l'**instance EC2** sur aws permettant le déploiement

Déploiement API

Annexe : API

Prêt à dépenser

Informations disponibles après modélisation

- **Données prêtes pour le modèle**
- **Données post-feature engineering non standardisées**
- **Données originales**
- **modèle entraîné**

API

- Télécharge les informations disponibles
- Crée un **dataset création graphs** dashboard réunisant les variables utilisées lors de la modélisation MAIS non standadisées et des variables apportant des informations génerales des clients (**/dataGeneral**).
- Met à disposition le **dataset pour la modélisation** (**/dataApi**).
- À partir de l'id client et du modèle entraînné, **calcule le score** (probabilité de défaillance), **classe le client**, calcule l'**importance des features** nécessaires pour la communication avec les clients (**/client/<idclient>**).

Flask

page d'accueil
/

/dataAPI → **/client/<idclient>**

/dataGeneral