PROJET 7

IMPLÉMENTEZ UN MODÈLE DE SCORING

Présenté par Kilian Alliot

10/08/2023

Plan

Présentation 01

Objectifs 02

Analyse 03

Modélisation 04

Déploiement 05

Data Drift 06

Démonstration 07

Présentation

Prêt à dépenser

Je suis Data Scientist au sein d'une société financière, nommée "Prêt à dépenser", qui propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt.

Objectifs

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

De plus, les chargés de relation client ont fait remonter le fait que les clients sont de plus en plus demandeurs de transparence vis-à-vis des décisions d’octroi de crédit. Cette demande de transparence des clients va tout à fait dans le sens des valeurs que l’entreprise veut incarner.

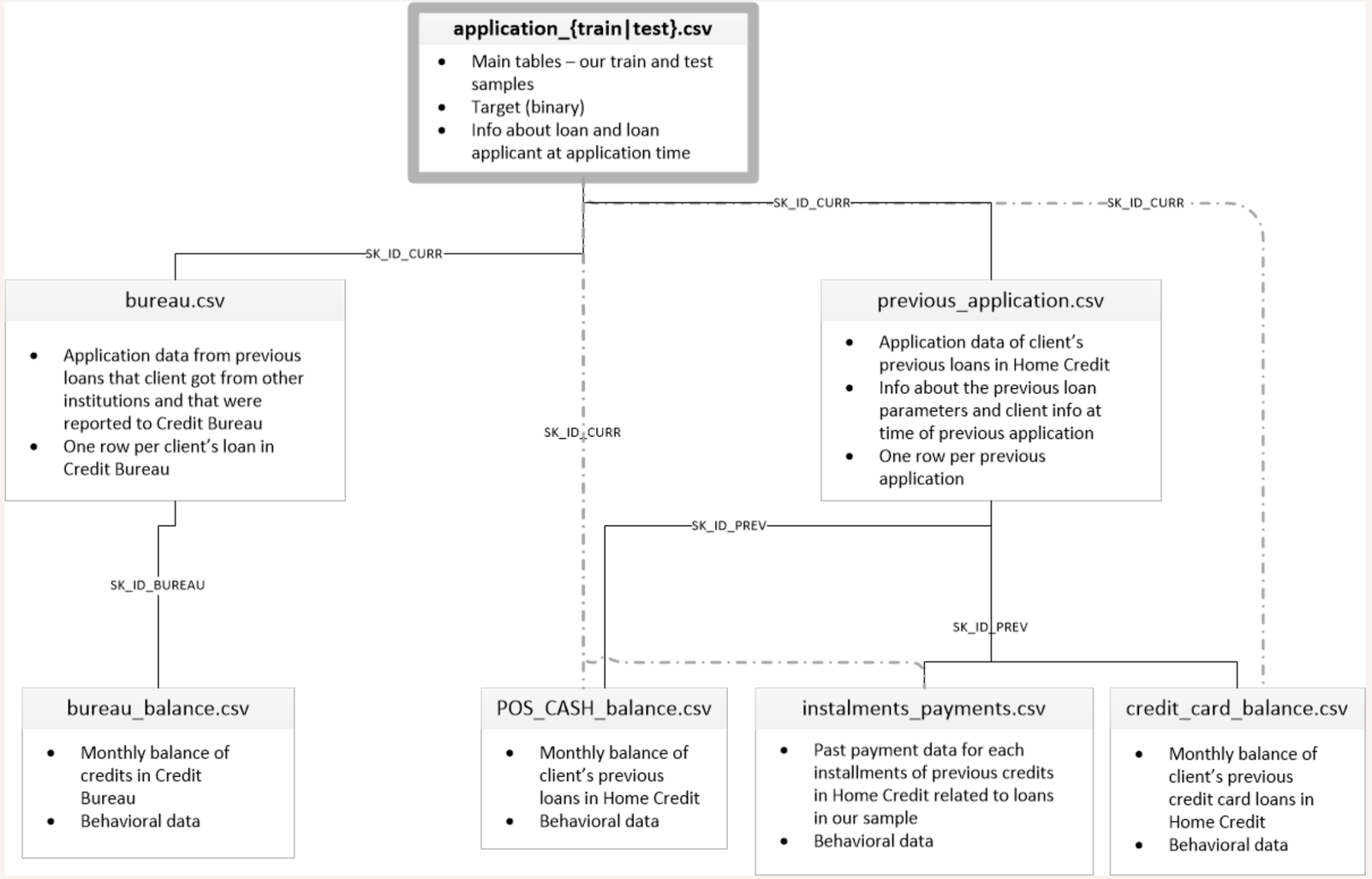
Prêt à dépenser décide donc de développer un dashboard interactif pour que les chargés de relation client puissent à la fois expliquer de façon la plus transparente possible les décisions d’octroi de crédit, mais également permettre à leurs clients de disposer de leurs informations personnelles et de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

Analyse

```
Training Data Shape: (307511, 122)
Testing Data Shape: (48744, 121)
```

App_train : 307k lignes et122 colonnes
App_test : 49kk lignes et121 colonnes

Tous les autres fichiers csv sont complémentaires avec app_train.
Ils apportent des informations supplémentaires sur le client.
Par exemple, les précédents crédits contractés et l'historique de leurs remboursements.

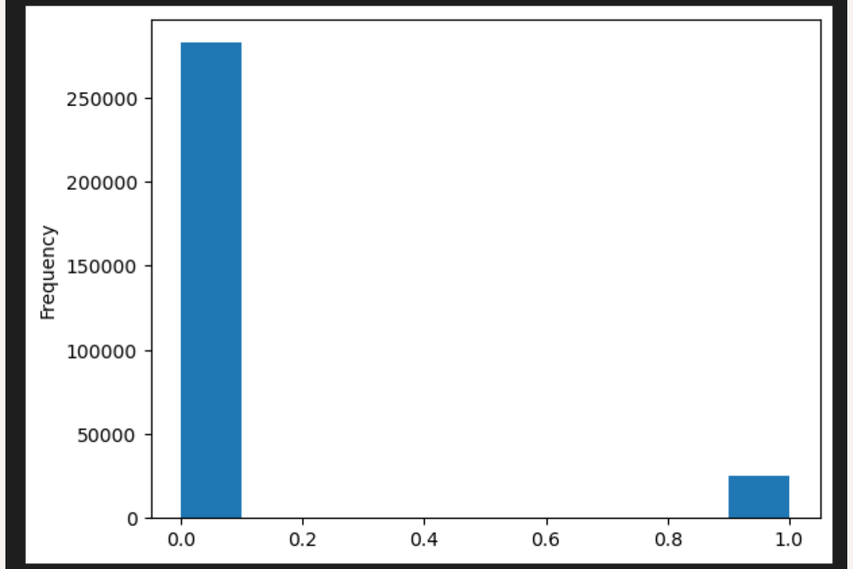


Analyse

Probabilité que le client puisse rembourser le prêt (noté 0)
Probabilité que le client soit en défaut de paiement et ne puisse pas rembourser le prêt (noté 1)

```
TARGET
0    282686
1     24825
dtype: int64

<Axes: ylabel='Frequency'>
```




```
Your selected dataframe has 122 columns.
There are 67 columns that have missing values.
```

	Missing Values	% of Total Values
COMMONAREA_MEDI	214865	69.9
COMMONAREA_AVG	214865	69.9
COMMONAREA_MODE	214865	69.9
NONLIVINGAPARTMENTS_MEDI	213514	69.4
NONLIVINGAPARTMENTS_MODE	213514	69.4

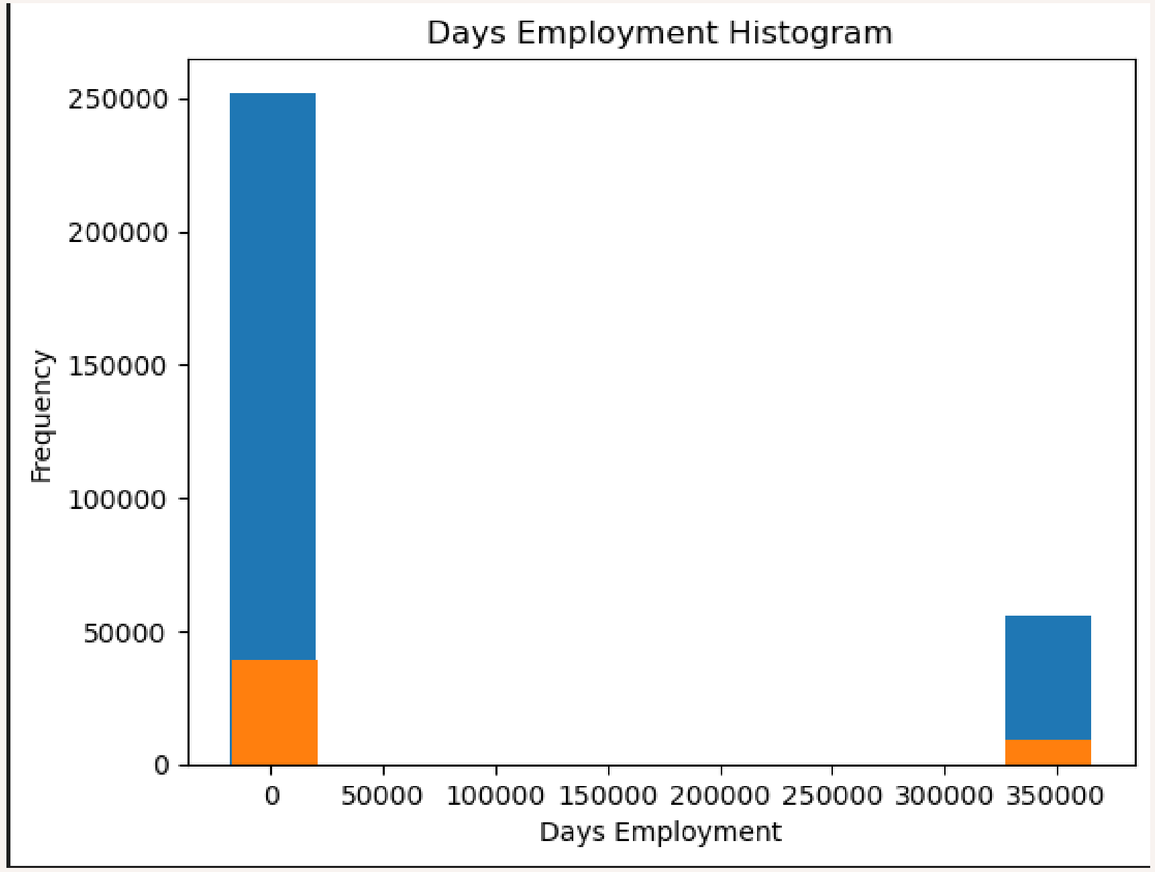
De nombreuses colonnes de notre jeu de données étaient peu remplies et peu pertinentes, elles ont été supprimées.
Pour le traitement des valeurs manquantes, la stratégie générale utilisée a été l'imputation par la médiane.
Les variables catégorielles ont été one hot encodées.

```
# one-hot encoding of categorical variables
app_train = pd.get_dummies(app_train)
app_test = pd.get_dummies(app_test)

print('Training Features shape: ', app_train.shape)
print('Testing Features shape: ', app_test.shape)
```

```
Training Features shape:  (307511, 217)
Testing Features shape:  (48744, 213)
```

Analyse



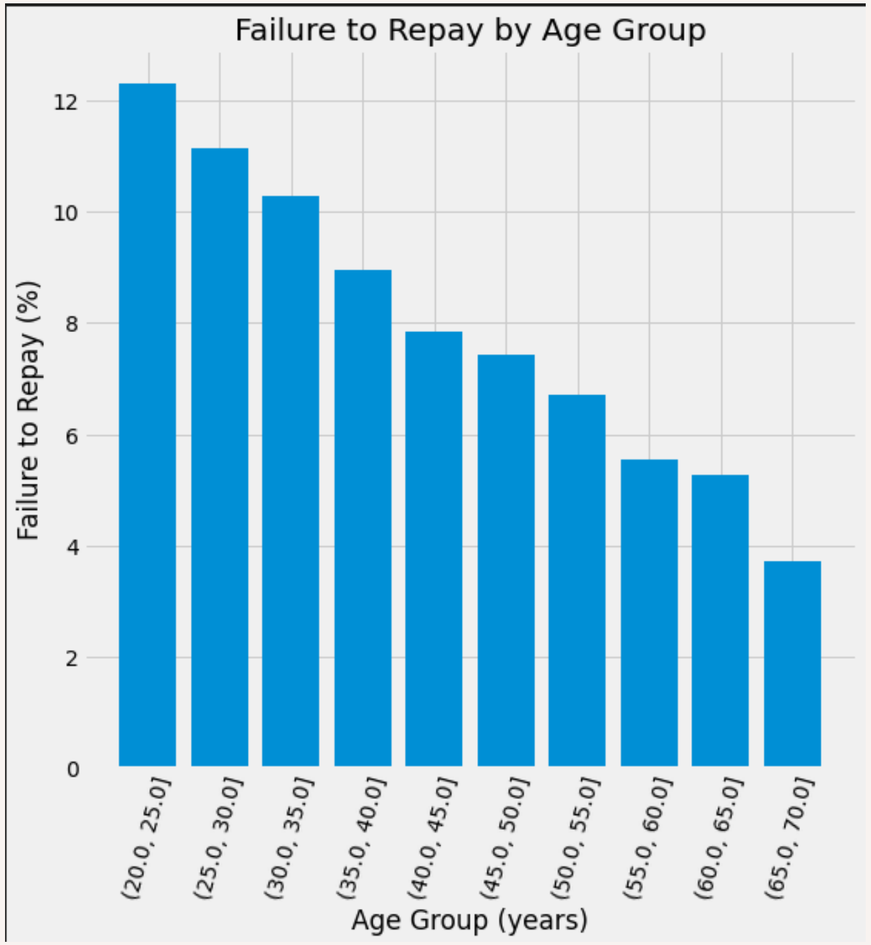
Les anomalies constatées au niveau de la variable 'DAYS_EMPLOYED' ont été traitées comme des valeurs manquantes.

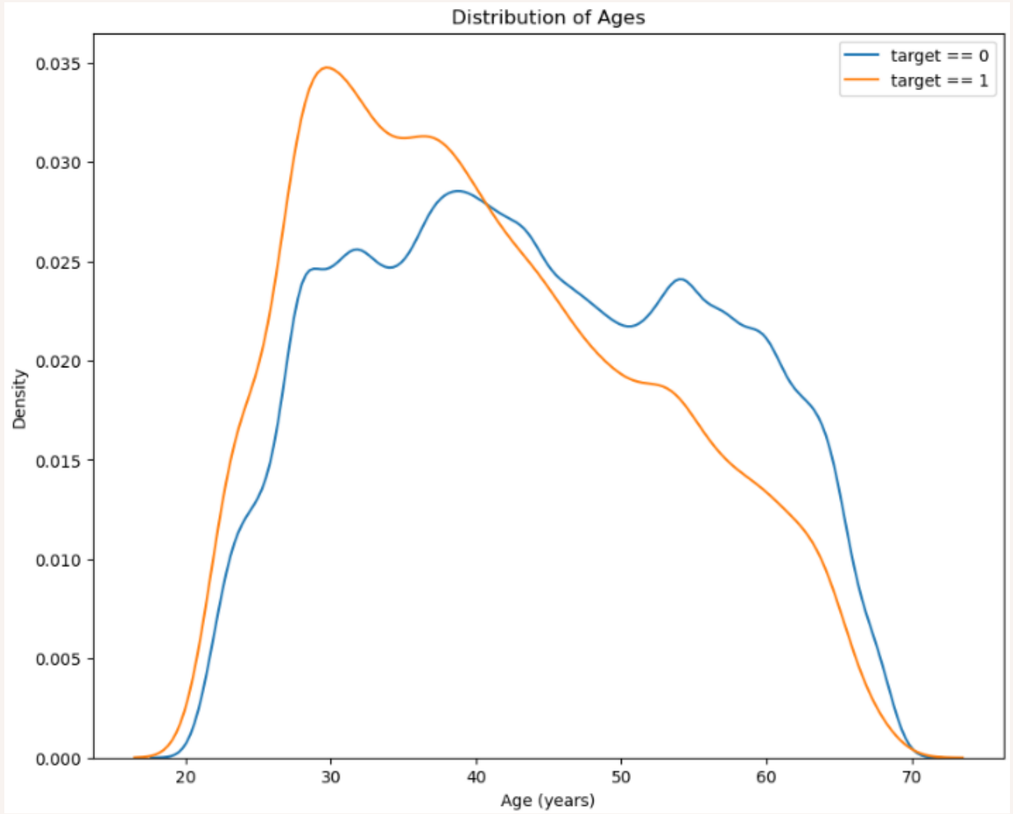
Corrélations

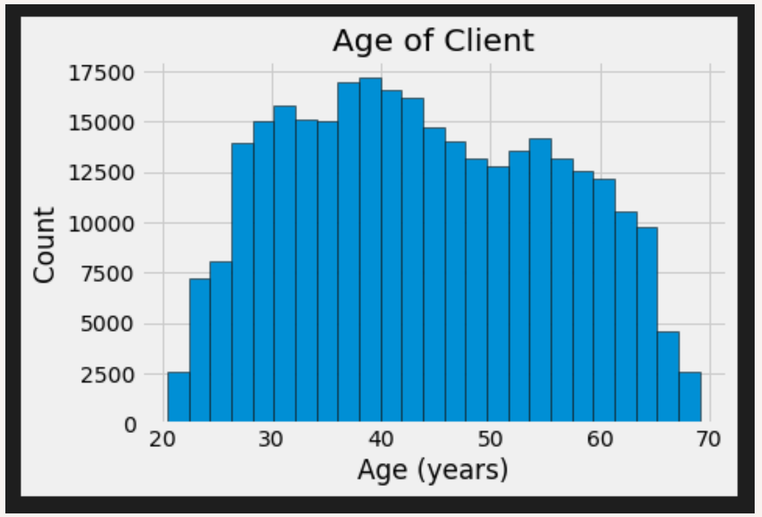
```
Most Positive Correlations:
 OCCUPATION_TYPE_Laborers                          0.043019
FLAG_DOCUMENT_3                                    0.044346
REG_CITY_NOT_LIVE_CITY                             0.044395
FLAG_EMP_PHONE                                     0.045982
NAME_EDUCATION_TYPE_Secondary / secondary special  0.049824
REG_CITY_NOT_WORK_CITY                             0.050994
DAYS_ID_PUBLISH                                    0.051457
CODE_GENDER_M                                      0.054713
DAYS_LAST_PHONE_CHANGE                             0.055218
NAME_INCOME_TYPE_Working                           0.057481
REGION_RATING_CLIENT                               0.058899
REGION_RATING_CLIENT_W_CITY                        0.060893
DAYS_EMPLOYED                                      0.074958
DAYS_BIRTH                                         0.078239
TARGET                                             1.000000
Name: TARGET, dtype: float64
```

```
Most Negative Correlations:
 EXT_SOURCE_3                                      -0.178919
EXT_SOURCE_2                                       -0.160472
EXT_SOURCE_1                                       -0.155317
NAME_EDUCATION_TYPE_Higher education               -0.056593
CODE_GENDER_F                                      -0.054704
NAME_INCOME_TYPE_Pensioner                         -0.046209
DAYS_EMPLOYED_ANOM                                 -0.045987
ORGANIZATION_TYPE_XNA                              -0.045987
FLOORSMAX_AVG                                      -0.044003
FLOORSMAX_MEDI                                     -0.043768
FLOORSMAX_MODE                                     -0.043226
EMERGENCYSTATE_MODE_No                             -0.042201
HOUSETYPE_MODE_block of flats                      -0.040594
AMT_GOODS_PRICE                                    -0.039645
REGION_POPULATION_RELATIVE                         -0.037227
Name: TARGET, dtype: float64
```

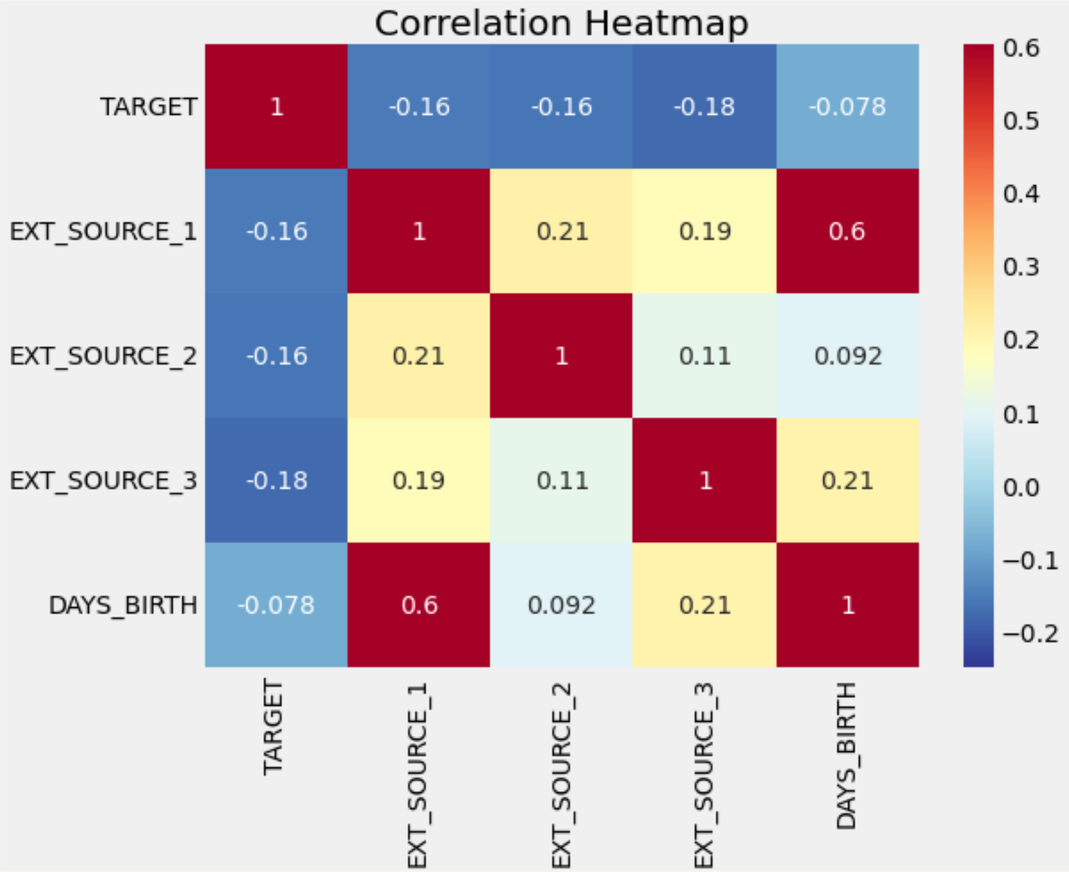
Analyse

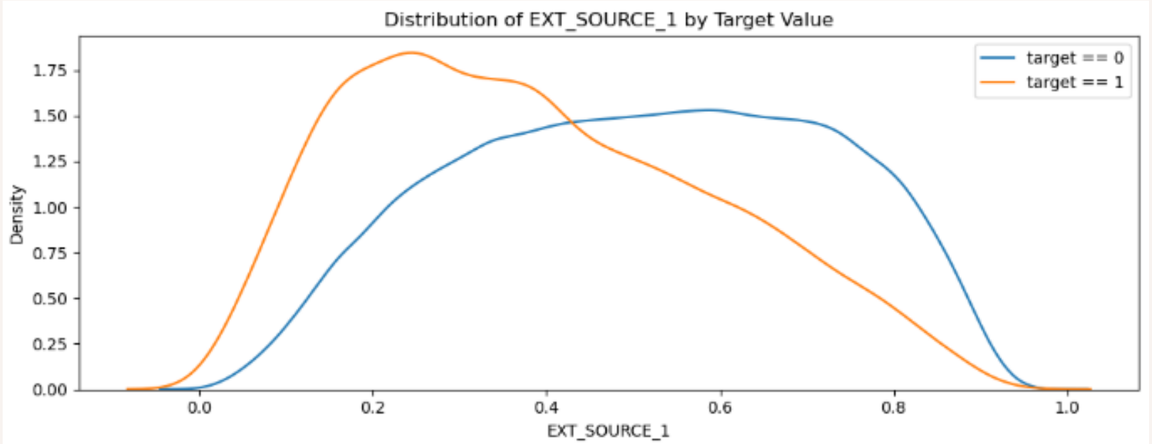

Failure to Repay by Age Group
Failure to Repay (%)
0
2
4
6
8
10
12
(20.0, 25.0]
(25.0, 30.0]
(30.0, 35.0]
(35.0, 40.0]
(40.0, 45.0]
(45.0, 50.0]
(50.0, 55.0]
(55.0, 60.0]
(60.0, 65.0]
(65.0, 70.0]
Age Group (years)

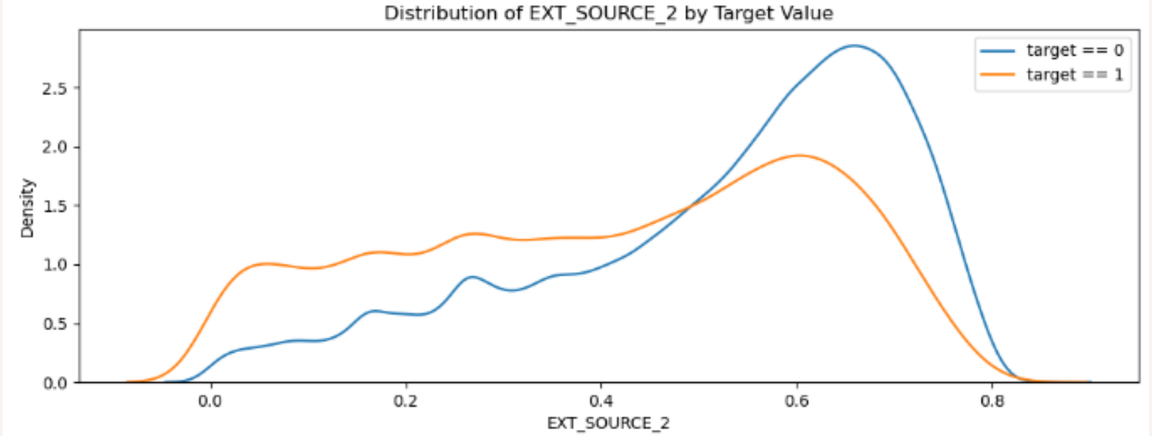

Distribution of Ages
target == 0
target == 1
Density
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
20
30
40
50
60
70
Age (years)

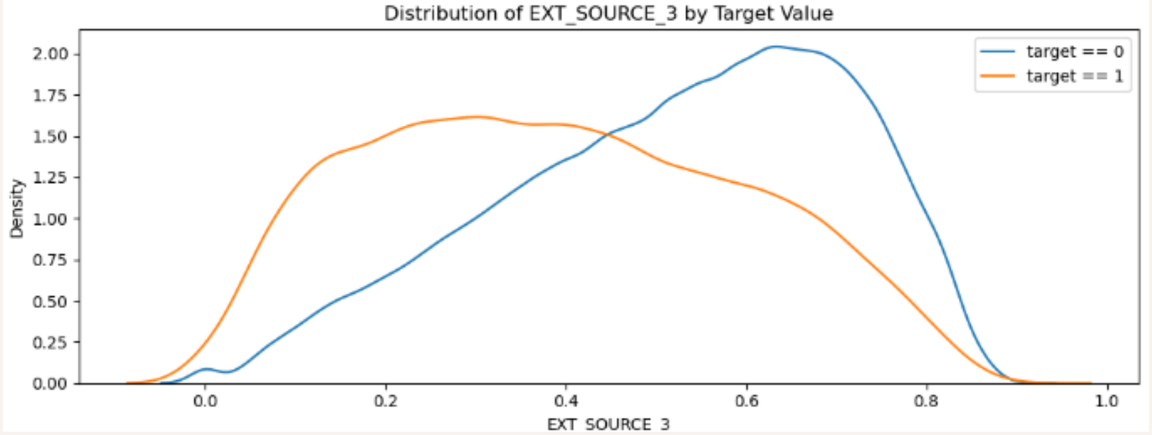

Age of Client
Count
0
2500
5000
7500
10000
12500
15000
17500
20
30
40
50
60
70
Age (years)

Analyse


Correlation Heatmap
TARGET
EXT_SOURCE_1
EXT_SOURCE_2
EXT_SOURCE_3
DAYS_BIRTH
1
-0.16
-0.16
-0.18
-0.078
-0.16
1
0.21
0.19
0.6
-0.16
0.21
1
0.11
0.092
-0.18
0.19
0.11
1
0.21
-0.078
0.6
0.092
0.21
1
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2


Distribution of EXT_SOURCE_1 by Target Value
target == 0
target == 1
Density
EXT_SOURCE_1


Distribution of EXT_SOURCE_2 by Target Value
target == 0
target == 1
Density
EXT_SOURCE_2


Distribution of EXT_SOURCE_3 by Target Value
target == 0
target == 1
Density
EXT_SOURCE_3

Analyse

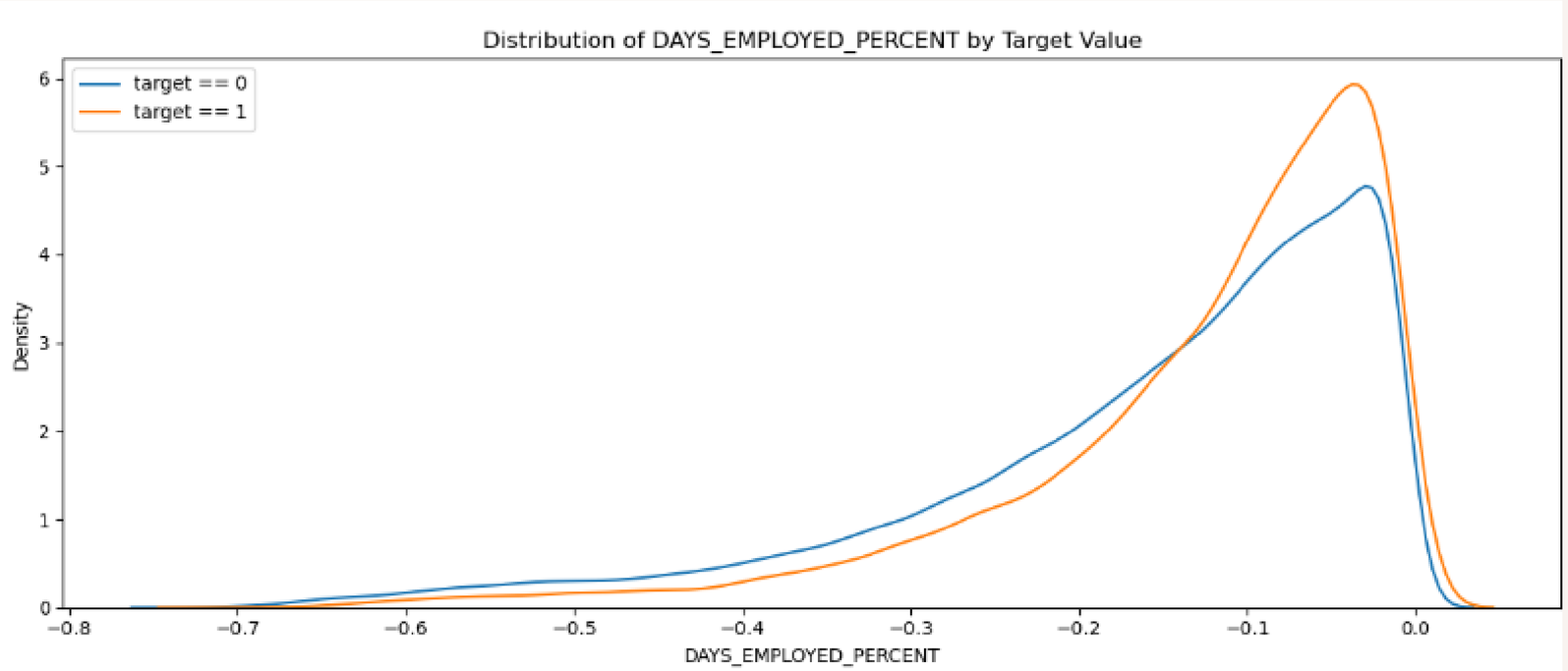
```
Training data with polynomial features shape:  (307511, 249)
Testing data with polynomial features shape:   (48744, 249)
```

Domain Knowledge Features

- CREDIT_INCOME_PERCENT: the percentage of the credit amount relative to a client's income
- ANNUITY_INCOME_PERCENT: the percentage of the loan annuity relative to a client's income
- CREDIT_TERM: the length of the payment in months (since the annuity is the monthly amount due
- DAYS_EMPLOYED_PERCENT: the percentage of the days employed relative to the client's age

Au niveau du 'Feature Engineering', trois stratégies ont été abodées :

- Utilisation du Dataset en l'état
- Polynomial features : création de plusieurs features polynomiales à partir des features les plus importantes (celles qui ont une corrélation importante avec la variable cible)
- Domain knowledge features : création de quatre features liées au domaine fonctionnel.



Modélisation

```
# logistic reg cross validate
{'fit_time': array([26.64608026, 26.87596512, 12.91484284, 20.69194603, 25.04282022]),
 'score_time': array([0.75971341, 0.20326853, 0.21254563, 0.20427799, 0.21911764]),
 'test_score': array([0.63045007, 0.64654634, 0.6513145 , 0.65986953, 0.60752373]),
 'train_score': array([0.75662333, 0.75818348, 0.73804827, 0.75032474, 0.75789132])}
```

```
# lgbm cross validate avec std scaler
{'fit_time': array([5.12360835, 2.37041664, 2.41124582, 2.05963945, 1.90335631]),
 'score_time': array([0.20288706, 0.60501051, 0.2047143 , 0.19802332, 0.19889545]),
 'test_score': array([0.76164244, 0.68518519, 0.70971522, 0.72366108, 0.66385406]),
 'train_score': array([1., 1., 1., 1., 1.])}
```

```
# rfc cross validate avec std scaler
{'fit_time': array([1.70818114, 1.21364379, 1.19071674, 1.20064211, 1.18891668]),
 'score_time': array([0.21861148, 0.21948099, 0.20915508, 0.20816493, 0.20764995]),
 'test_score': array([0.74281138, 0.69882013, 0.67410836, 0.73157155, 0.69891402]),
 'train_score': array([1., 1., 1., 1., 1.])}
```

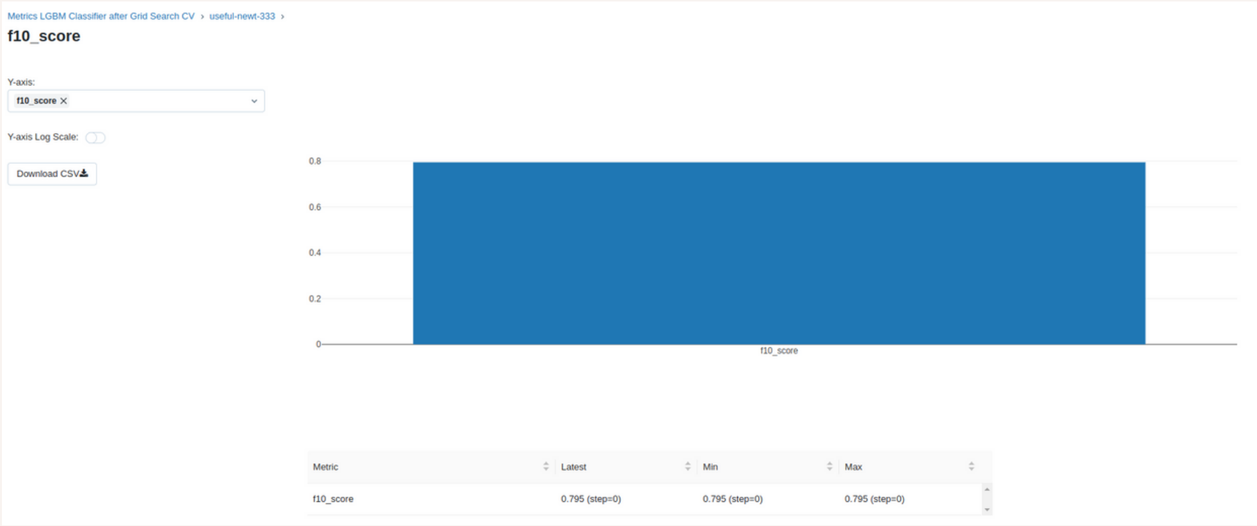
Pour la modélisation, j'ai créé une pipeline comprenant un column transformer, un scaler et un classifier.
Dataset est composé le 80k individus, 60k de classe 0 et 20k de classe 1.
Le test size est de 0.2 (64k-16k).
Ces 3 modèles ont été testés avec un StandardScaler et quelques hyperparamètres différents afin d'avoir rapidement des résultats.
LGBM étant rapide et prometteur, j'ai choisi de continuer avec lui.

Modélisation

Continuons donc avec LGBM.
Dans le cadre du projet, j'ai utilisé le paramètre class_weigh qui attribue des poids différents aux individus afin de rééquilibrer cette différence de population lors de l'entraînement.
Le choix du meilleurs modèle a été effectué en retenant le modèle ayant obtenu le meilleurs score sur le jeu de validation.
Dans le cadre du projet, on peut supposer, par exemple, que le coût d'un FN est dix fois supérieur au coût d'un FP. C'est pourquoi j'ai mis en place un score «métier» approprié,
appelé f10_score. Ce score reprend le f_beta_score avec un facteur 10.
En effet, on cherche à limiter le nombre de False Negative car cela est une perte financière pour l'entreprise, et on cherche aussi à limiter le nombre de FP car ça représente une perte de chiffre d'affaire pour l'entreprise.
Le scoring a optimisé est à deux composantes : scoring = {"AUC": "roc_auc", "my_f10_score": f10_score}

```
param_grid_lgbm = {
        'preprocessor__num_scaler': [StandardScaler(), MinMaxScaler(), RobustScaler()],
        'classifier__max_depth': [-1, 10],
        'classifier__learning_rate': [0.01, 0.1],
        'classifier__n_estimators': [100, 1000, 2000],
        'classifier__num_leaves': [30, 50],
        'classifier__class_weight': ['balanced'],
    }
```

Modélisation




```
Params {'classifier__class_weight': 'balanced', 'classifier__learning_rate': 0.01, 'classifier__max_depth': -1, 'classifier__n_estimators': 100, 'classifier__num_leaves': 30, 'preprocessor__num_scaler': MinMaxScaler()}
mean_train_AUC 0.7676080686199835
mean_test_AUC 0.7479241313403103
mean_train_my_f10_score 0.6985285728698163
mean_test_my_f10_score 0.6772122514563985

Params {'classifier__class_weight': 'balanced', 'classifier__learning_rate': 0.01, 'classifier__max_depth': -1, 'classifier__n_estimators': 100, 'classifier__num_leaves': 30, 'preprocessor__num_scaler': RobustScaler()}
mean_train_AUC 0.7677268540653813
mean_test_AUC 0.7483784047949482
mean_train_my_f10_score 0.6974640425363751
mean_test_my_f10_score 0.6780425962908948

Params {'classifier__class_weight': 'balanced', 'classifier__learning_rate': 0.01, 'classifier__max_depth': -1, 'classifier__n_estimators': 100, 'classifier__num_leaves': 50, 'preprocessor__num_scaler': StandardScaler()}
mean_train_AUC 0.7844090077079163
mean_test_AUC 0.752751413078016
mean_train_my_f10_score 0.7151558437246861
mean_test_my_f10_score 0.6769186774859403
```

Modélisation

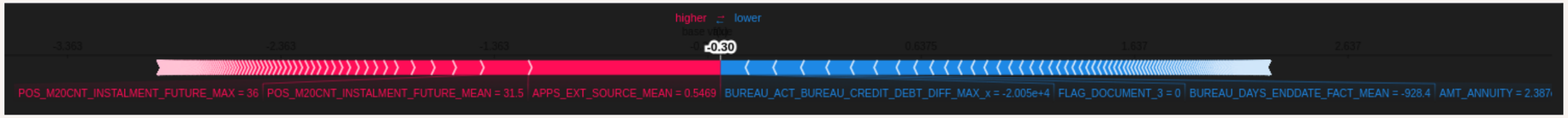
```
LGBMClassifier
LGBMClassifier(class_weight='balanced', learning_rate=0.01, n_estimators=1000,
               num_leaves=30)
```

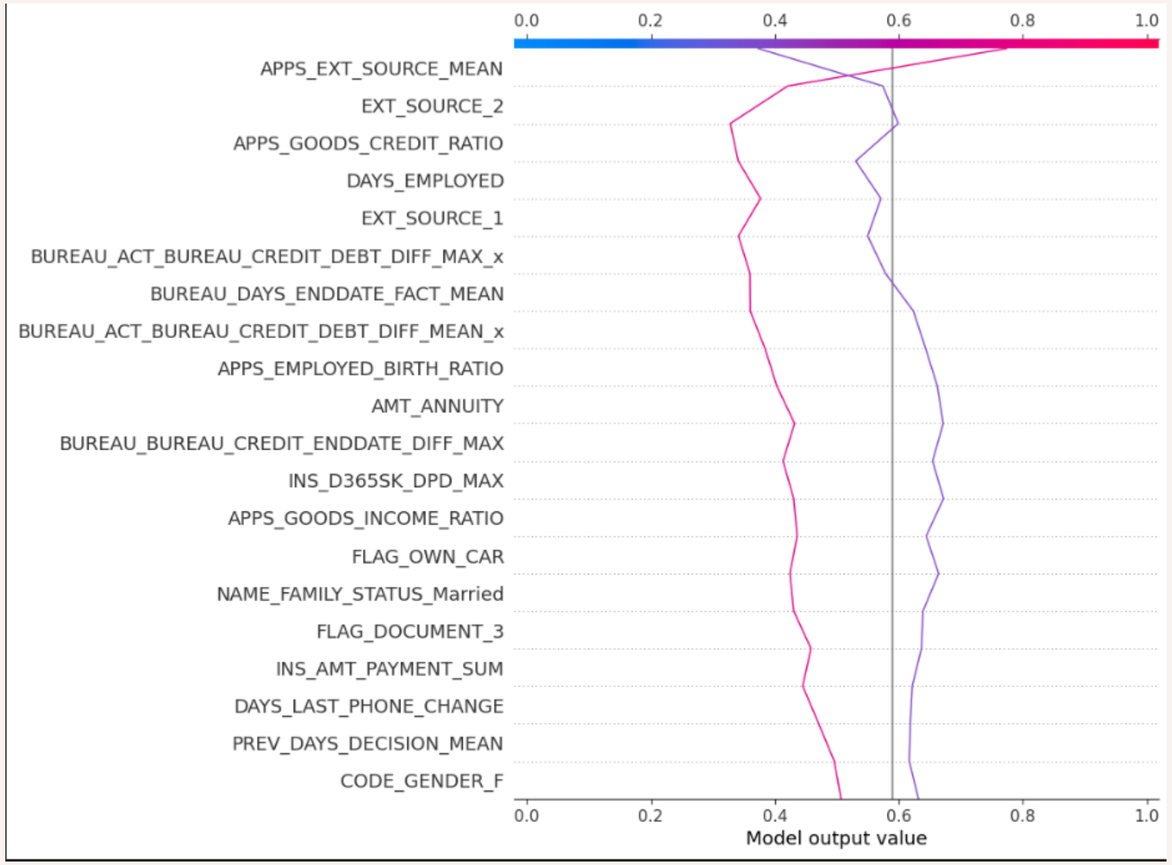
Tableau de synthèse de la modélisation

	AUC_train_set	f_10_score_train_set	AUC_validation_set	f_10_score_validation_set	AUC_test_set	f_10_score_test_set
LGBMClassifier	1	1	0.78	0.68	0.71	0.69

	Feature_importances_score	Col_name
0	962	APPS_ANNUITY_CREDIT_RATIO
1	890	APPS_EXT_SOURCE_MEAN
2	727	DAYS_BIRTH
3	569	AMT_ANNUITY
4	555	APPS_GOODS_CREDIT_RATIO
5	468	EXT_SOURCE_1
6	464	EXT_SOURCE_2
7	442	INS_D365SK_DPD_MAX
8	433	POS_M20CNT_INSTALMENT_FUTURE_MEAN
9	406	DAYS_ID_PUBLISH

Feature Importance





Déploiement

Cliquez sur l'cône pour accéder au projet

MyNameIsKilian / **OC-Projet-7**

Type / to search

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

OC-Projet-7 Public

Pin | Unwatch 1 | Fork 0 | Star 0

main | **2** branches | **0** tags | Go to file | Add file | Code

Kilian Fix tests with good relative path — 8874661 20 minutes ago — **71** commits

.github/workflows	Clean up tests part 3	1 hour ago
api	Define clean project architecture	4 days ago
dashboard	Define clean project architecture	4 days ago
data	Define clean project architecture	4 days ago
mlruns/877697581142665801	Run GridSearch on big dataset	2 months ago
models	Define clean project architecture	4 days ago
notebooks	Clean up tests part 3	1 hour ago
tests	Fix tests with good relative path	20 minutes ago
.gitignore	Define clean project architecture	4 days ago
README.md	Update Readme file	4 days ago
requirements.txt	Define clean project architecture	4 days ago

README.md

OC - Projet 7 - Parcours Data Scientist

About

No description, website, or topics provided.

Readme
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

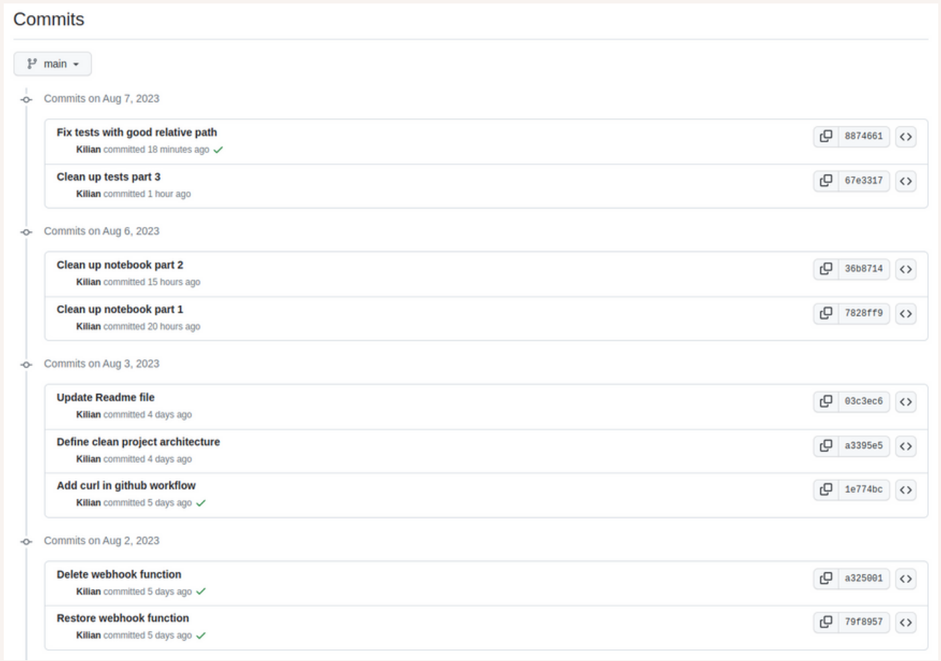
Languages

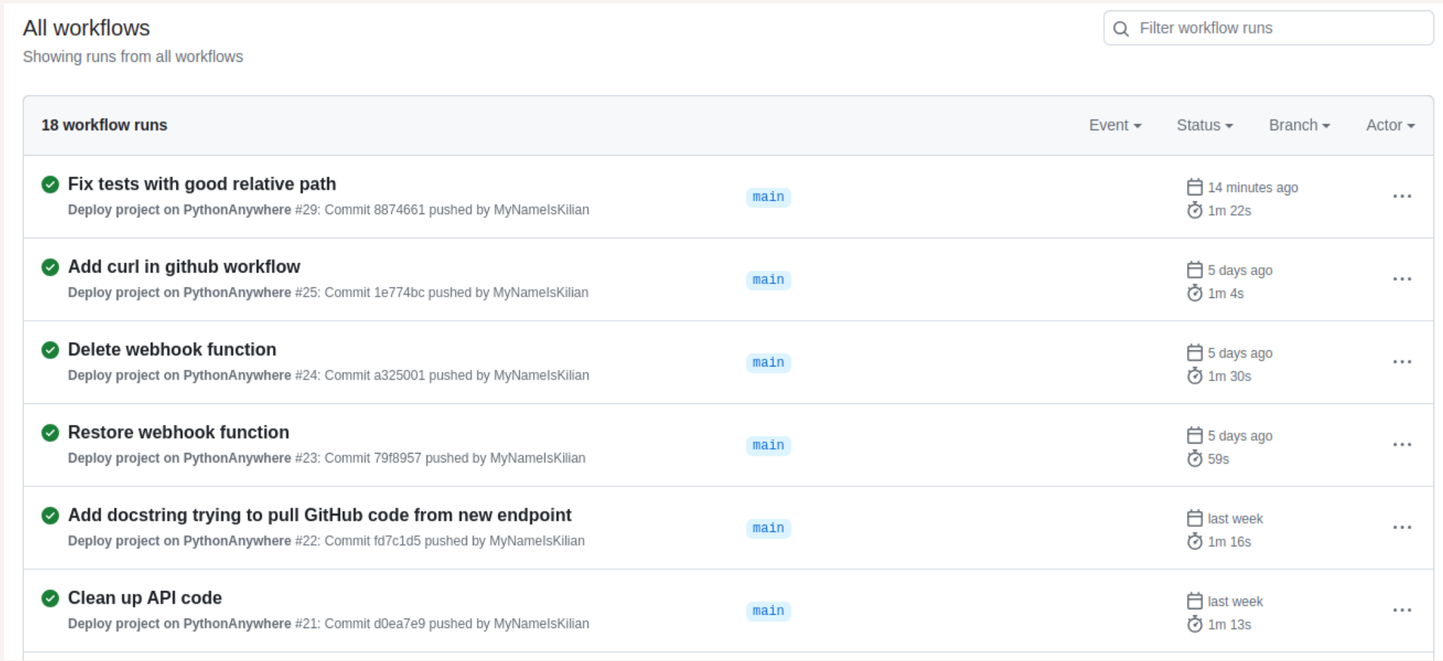
Jupyter Notebook 100.0%

Déploiement

J'utilise git qui est un outil de versionning de code. Le code de mon projet est envoyé sur mon GitHub avec la commande git push.
Lorsque du code nouveau est envoyé sur la branche main (remote), une pipeline GitHub action est actionnée.
Des tests unitaires sont lancés et lorsque tout est vérifié et fonctionne:

- le code à jour est ajouté à la branche main
- le repository hébergé sur Pythonanywhere exécute la commande git pull afin de récupérer le code à jour





Déploiement

build
succeeded 14 minutes ago in 1m 14s

Search logs

- Set up job — 2s
- Checkout code — 3s
- Set up Python 3.9 — 0s
- Install dependencies — 43s
- Run unit tests — 3s

```
1  ▶ Run pytest -v --disable-warnings
8  ============================= test session starts ==============================
9  platform linux -- Python 3.9.17, pytest-7.3.2, pluggy-1.2.0 -- /opt/hostedtoolcache/Python/3.9.17/x64/bin/python
10 cachedir: .pytest_cache
11 rootdir: /home/runner/work/OC-Projet-7/OC-Projet-7
12 collecting ... collected 4 items
13
14 tests/test_class.py::TestClass::test_add_numbers PASSED                [ 25%]
15 tests/test_class.py::TestClass::test_model_leaves PASSED               [ 50%]
16 tests/test_class.py::TestClass::test_predict_class PASSED              [ 75%]
17 tests/test_class.py::TestClass::test_expected_value_from_explainer PASSED [100%]
18
19 ========================= 4 passed, 20 warnings in 2.11s =========================
```

- Deploy to PythonAnywhere — 20s
- Post Set up Python 3.9 — 0s
- Post Checkout code — 0s
- Complete job — 0s

Data Drift

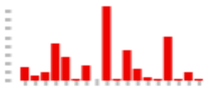
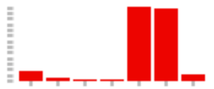
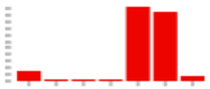
EVIDENTLY AI

On utilise la librairie Evidently pour détecter un éventuel data drift.
On compare ici app_train et app_test.
Aucun datadrift significatif n'est détecté.

Dataset Drift

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

17	1	0.0588
Columns	Drifted Columns	Share of Drifted Columns

Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
NAME_CONTRACT_TYPE	cat			Detected	Jensen-Shannon distance	0.14755
WEEKDAY_APPR_PROCESS_START	cat			Not Detected	Jensen-Shannon distance	0.039686
ORGANIZATION_TYPE	cat			Not Detected	Jensen-Shannon distance	0.026289
NAME_TYPE_SUITE	cat			Not Detected	Jensen-Shannon distance	0.020448
NAME_FAMILY_STATUS	cat			Not Detected	Jensen-Shannon distance	0.01978
OCCUPATION_TYPE	cat			Not Detected	Jensen-Shannon distance	0.017941
NAME_EDUCATION_TYPE	cat			Not Detected	Jensen-Shannon distance	0.01516
WALLSMATERIAL_MODE	cat			Not Detected	Jensen-Shannon distance	0.013779
NAME_INCOME_TYPE	cat			Not Detected	Jensen-Shannon distance	0.011543
NAME_HOUSING_TYPE	cat			Not Detected	Jensen-Shannon distance	0.011137

Démonstration

pythonanywhere

Streamlit

Cliquez sur l'image pour accéder au dashboard

Merci de votre attention