

Implémentez un modèle de scoring

...

Benoit Coppin

Plan

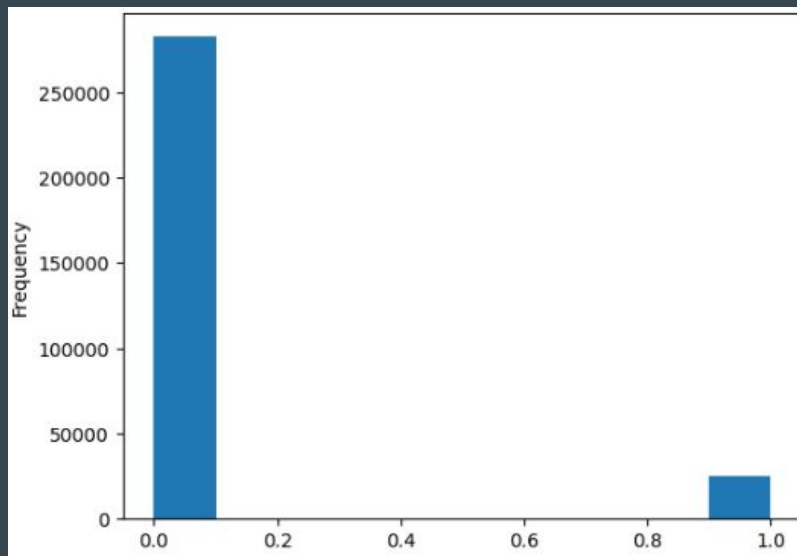
- Rappel des objectifs
- La base de données
- ML flow
- Les différents modèles
- Équilibre des classes
- Data Engineering
- Résultat Final
- Métrique métier
- Data Drift
- Test Unitaire
- Github
- Github Action

Rappel des objectifs

- Analyser et nettoyer une base de données
- Ajout de colonne par Features Engineering
- Entraîner des modèles pour faire un système de scoring
- Équilibrer le meilleur modèle pour limiter les faux Négatifs
- Réaliser une API
- Réaliser un dashboard interactif :
 - Visualiser le score et son interprétation
 - Comparer le client à l'ensemble des clients

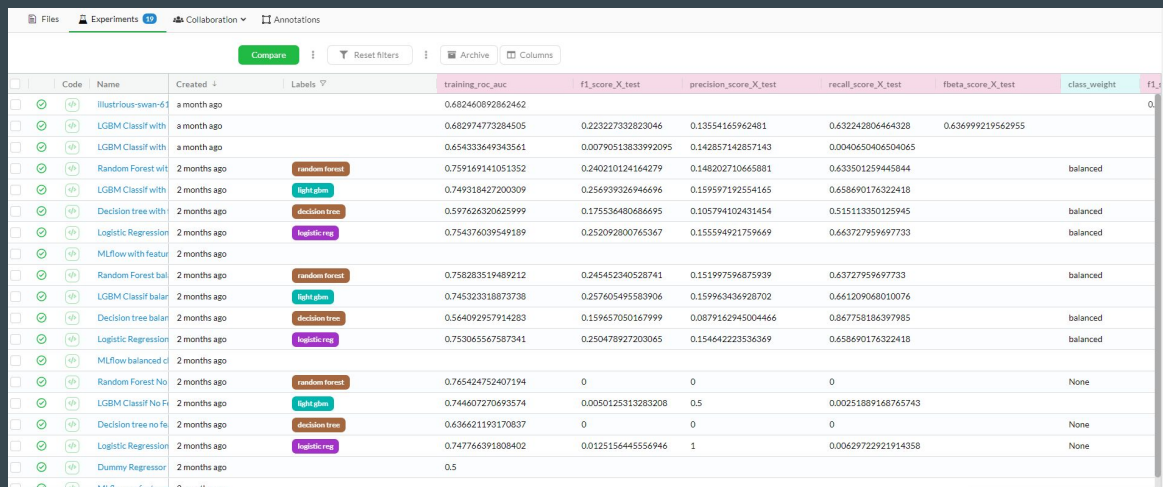
La base de données

- Caractéristique :
 - 307511 lignes et 122 colonnes
- Déséquilibre bon/mauvais client :
 - 282686 bons clients
 - 24825 mauvais clients
 - 24% de valeurs manquantes au total
 - Pas de doublon



ML Flow

- Mise en place de ml flow :
 - Sauvegarde des modèles
 - Sauvegarde des scores
- Relié à Dags Hub :
 - Base de données accessible de n'importe où



	Code	Name	Created	Labels	training_roc_auc	f1_score_X_test	precision_score_X_test	recall_score_X_test	fbeta_score_X_test	class_weight	f1_s
	✓	illustrious-swain-6	a month ago		0.682460892862462						0
	✓	LGBM Classif with	a month ago		0.682974773284505	0.223227332823046	0.13554165962481	0.632242806464328	0.636999219562955		
	✓	LGBM Classif with	a month ago		0.654333649343561	0.00790513833992095	0.142857142857143	0.0040650406504065			
	✓	Random Forest wit	2 months ago	random forest	0.759169141051352	0.240210124164279	0.148202710665881	0.633501259445844		balanced	
	✓	LGBM Classif with	2 months ago	light gbm	0.749318427200309	0.256939326946696	0.159597192554165	0.658690176322418			
	✓	Decision tree with	2 months ago	decision tree	0.597626320625999	0.175536480686695	0.105794102431454	0.515113350125945		balanced	
	✓	Logistic Regression	2 months ago	logistic reg	0.754376039549189	0.252092800765367	0.155594921759669	0.663727959697733		balanced	
	✓	MLflow with featur	2 months ago								
	✓	Random Forest bal	2 months ago	random forest	0.758283519489212	0.245452340528741	0.151997596875939	0.63727959697733		balanced	
	✓	LGBM Classif balanc	2 months ago	light gbm	0.745323318873738	0.257605495583906	0.159963436928702	0.661209068010076			
	✓	Decision tree balanc	2 months ago	decision tree	0.564092957914283	0.159657050167999	0.0879162945004466	0.867758186397985		balanced	
	✓	Logistic Regression	2 months ago	logistic reg	0.753065567587341	0.250478927203065	0.1546422323536369	0.658690176322418		balanced	
	✓	MLflow balanced cl	2 months ago								
	✓	Random Forest No	2 months ago	random forest	0.765424752407194	0	0	0		None	
	✓	LGBM Classif No Fi	2 months ago	light gbm	0.744607270693574	0.0050125313283208	0.5	0.00251889168765743			
	✓	Decision tree no fe	2 months ago	decision tree	0.636621193170837	0	0	0		None	
	✓	Logistic Regression	2 months ago	logistic reg	0.747766391808402	0.0125156445556946	1	0.00629722921914358		None	
	✓	Dummy Regressor	2 months ago		0.5						

Quels modèles ?

- Plusieurs modèles testés pour le scoring :
 - Logistic Regression
 - Decision Tree
 - LightGBM
 - RandomForest

Logistic Regression

Logistic Regression optimise les paramètres du modèle en utilisant une technique appelée maximum de vraisemblance.

Ajuste les coefficients qui sont pondérés en fonction de leur influence sur la prédiction de la variable à prédire.

- **Avantage :**
 - Faible demande de ressources
 - Bonne performance si données bien séparées
- **Désavantage:**
 - Besoin de linéarité
 - Sensible données aberrantes

Decision tree

L'arbre de décision divise récursivement les données en fonction des caractéristiques pour prendre des décisions de classification en utilisant des règles conditionnelles simples.

- **Avantage :**
 - Interprétabilité facile
 - Gère les données manquantes sans traitement
- **Avantage :**
 - Tendance à avoir de l'overfitting
 - Ne capture pas bien les relations complexes (relation non linéaire)

LightGBM

LightGBM est un algorithme de boosting basé sur les arbres de décision qui utilise une stratégie de construction des arbres appelée 'leaf-wise' pour améliorer l'efficacité et les performances du modèle en sélectionnant les feuilles les plus prometteuses lors de la construction de l'arbre.

- **Avantage :**
 - Haute Performance
 - Gère les données déséquilibré
- **Désavantage:**
 - Sensible valeur aberrantes
 - Nécessite beaucoup de données

Random Forest

Random Forest est un algorithme qui construit un ensemble de plusieurs arbres de décision et utilise l'agrégation des prédictions pour améliorer la performance de prédiction.

- **Avantage :**
 - Robuste contre les valeurs aberrantes
- **Désavantage :**
 - Difficile à interpréter
 - Beaucoup de ressources nécessaires

Les mesures

- ROC AUC : la capacité d'un modèle de classification à discriminer entre les classes
- Precision : mesure la proportion d'exemples positifs correctement classés par rapport à tous les exemples classés comme positifs
- Recall : mesure la proportion d'exemples positifs correctement classés par rapport à tous les exemples réellement positifs dans l'ensemble de données. Il quantifie la capacité du modèle à identifier tous les exemples positifs, sans se soucier des faux positifs.
- F1 score : combine la précision et le rappel en une seule valeur, permettant de mesurer l'équilibre entre la capacité du modèle à identifier correctement les exemples positifs et à minimiser les faux positifs et les faux négatifs.

Premier Résultat

	ROC AUC	f1 score	<u>precision</u>	recall
Logistic Reg	0.747	0.0125	1	0.006
Random forest	0.765	0	0	0
Decision tree	0.636	0	0	0
LightGBM	0.744	0.005	0.5	0.0025

Très mauvais f1 Score, Precision Score et Recall Score

S'explique de part le manque d'équilibre entre le nombre de bon client et mauvais client

Equilibre des Classes

Utilisation du paramètre 'class_weight' pour équilibre le nombre de bons et mauvais clients

Donne un poids à chaque classe pour donner plus d'importance au classe sous représenté et aidera à la performance des modèles

Résultat avec équilibre

	ROC AUC	f1 score	<u>precision</u>	recall
Logistic Reg	0.743	0.25	0.154	0.658
Random forest	0.758	0.245	0.151	0.637
Decision tree	0.564	0.159	0.087	0.867
LightGBM	0.745	0.257	0.15	0.661

Amélioration des scores, F1, Precision et Recall

Features Engineering

Création de nouvelles variables (combinaison de variable déjà existante) pour aider les modèles de machine Learning

- Ajout des colonnes :
 - CREDIT_INCOME_PERCENT
 - ANNUITY_INCOME_PERCENT
 - CREDIT_TERM
 - DAYS_EMPLOYED_PERCENT

Résultat Data Engineering

	ROC AUC	f1 score	<u>precision</u>	recall
Logistic Reg	0.744	0.252	0.155	0.660
Random forest	0.759	0.240	0.148	0.633
Decision tree	0.597	0.175	0.105	0.51
LightGBM	0.749	0.256	0.159	0.658

Avec l'ajout de ces colonnes, nous avons eu une faible augmentation des résultats

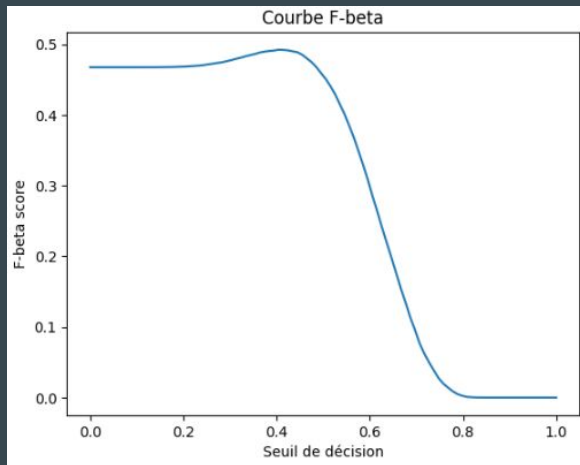
Notre meilleur modèle est donc le modèle LIGHTGBM

Métrique métier

Les faux négatifs ont un coup 10 à 20 fois supérieurs à un faux positif nous souhaitons donc les limiter.

- Comment limiter les faux négatifs ?
 - Utiliser le FBeta score :
 - Il combine le precision et recall score
 - Nous allons mettre un score supérieur à 1 :
 - Focus sur le Recall score (faux négatifs)
 - Définition d'un seuil de décision pour renforcer la limitation

Seuil de décision



Utilisation de la courbe F-beta:

Seuil optimal : 0.404

Matrice de confusion avec seuil de décision 0.5:

```
[[181297 101389]
```

```
 [ 9314 15511]]
```

Matrice de confusion avec seuil de décision optimal:

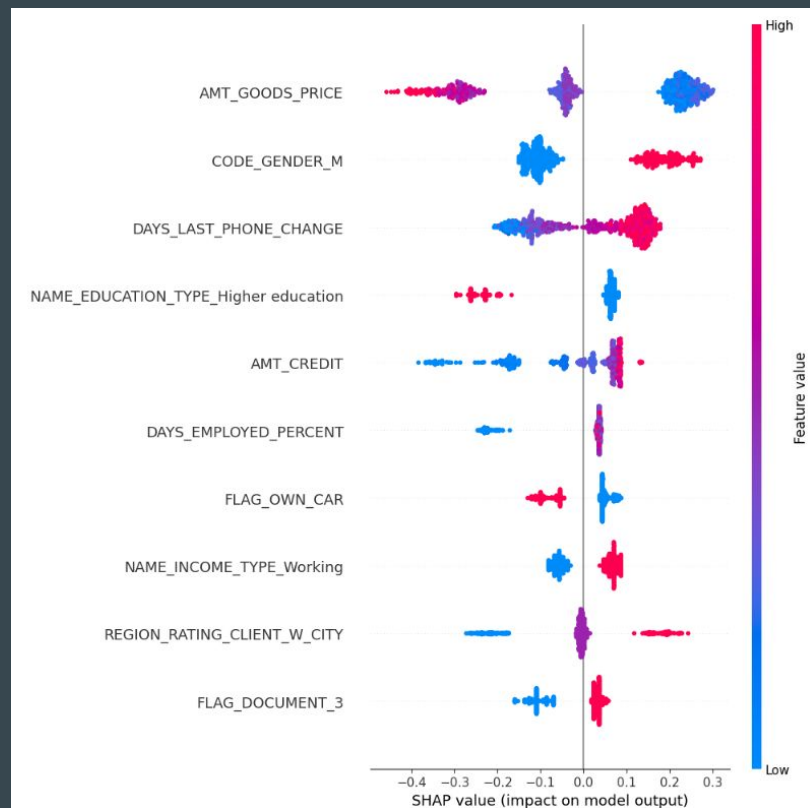
```
[[ 92473 190213]
```

```
 [ 3405 21420]]
```

Nous pouvons observer une nette diminution des faux négatifs

Interpretation modèle

- Top 5 Features :
 - AMT_GOODS_PRICE
 - CODE_GENDER_M
 - DAYS_LAST_PHONE_CHANGE
 - NAME_EDUCATION_TYPE
 - AMT_CREDIT



Data drift

Le Data Drift est la variation des propriétés des données au fil du temps, ce qui peut affecter les performances des modèles d'apprentissage automatique.

Nous allons utiliser l'outil Evidently qui permet de détecter les variations

Data drift

Seulement 6% des colonnes sont détectées comme 'Drifted'.

Les colonnes contenant du data drift sont parmi les 40 features les moins importantes pour notre modèle de machine Learning

Dataset Drift

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

234

Columns

14

Drifted Columns

0.0598

Share of Drifted Columns

Test Unitaire

Les tests unitaires sont des petites unités de code qui vérifient le bon fonctionnement d'une partie spécifique d'un programme ou d'une fonction

- Dans notre cas :
 - Vérifie le score pour un bon et mauvais client

```
import unittest
import json
from API.Fonction_Test import traitement
import lightgbm as lgb
import pandas as pd

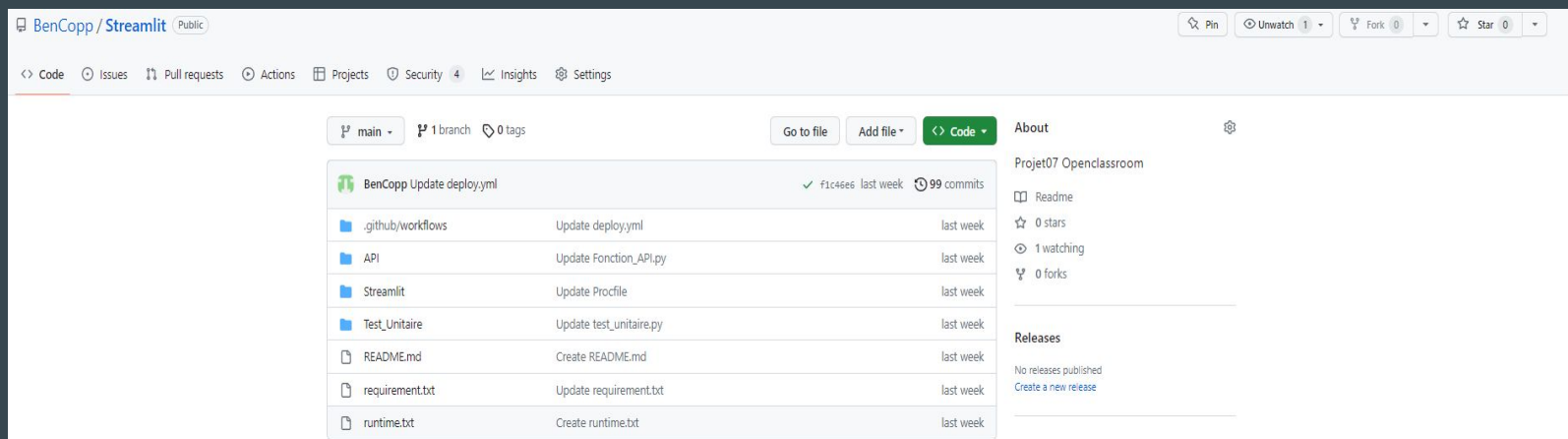
class MyTest(unittest.TestCase):
    def test_prediction_good(self):
        model = lgb.Booster(model_file='API/my_model.txt')
        df = pd.read_csv('Streamlit/df.csv')
        df_pred = df.loc[df['SK_ID_CURR'] == 156685]
        df_dict = df_pred.to_dict()
        df = traitement(df_dict)
        y_pred = model.predict(df)
        y_pred = y_pred.reshape((-1, 1))
        expected_value = 0.217
        self.assertAlmostEqual(y_pred[0][0], expected_value, places=3)

    def test_prediction_bad(self):
        model = lgb.Booster(model_file='API/my_model.txt')
        df = pd.read_csv('Streamlit/df.csv')
        df_pred = df.loc[df['SK_ID_CURR'] == 389871]
        df_dict = df_pred.to_dict()
        df = traitement(df_dict)
        y_pred = model.predict(df)
        y_pred = y_pred.reshape((-1, 1))
        expected_value = 0.621
        self.assertAlmostEqual(y_pred[0][0], expected_value, places=3)

if __name__ == '__main__':
    unittest.main()
```

GitHub

GitHub est une plateforme de développement collaboratif basée sur Git, qui permet de travailler sur des projets logiciels. Il offre des fonctionnalités telles que le contrôle de version, la gestion des problèmes, le déploiement, etc...



Github Action

GitHub Actions est un service d'intégration qui permet d'automatiser des tâches et des workflows personnalisés dans votre dépôt GitHub, tels que l'exécution de tests, le déploiement de votre application, etc.

- Dans notre cas :
 - Lors d'un push, lance le test unitaire et si celui-ci fonctionne, déploie l'API et le Dashboard interactif sur Heroku

```
name: Deploy

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest
    steps:

      - uses: actions/checkout@v2

      - name: Upgrade pip
        run: pip install --upgrade pip

      - name: Install dependencies
        run: pip install -r requirement.txt

      - name: Run unit tests
        run: python -m unittest discover Test_Unitaire

      - uses: akhileshns/heroku-deploy@v3.12.12 # This is the action
        with:
          heroku_api_key: ${secrets.HEROKU_API_KEY}
          heroku_app_name: "api-open-classroom" #Must be unique in Heroku
          heroku_email: "benoit.cppn@gmail.com"
          appdir: "API" #

      - uses: akhileshns/heroku-deploy@v3.12.12 # This is the action
        with:
          heroku_api_key: ${secrets.HEROKU_API_KEY}
          heroku_app_name: "dashboard-open-classroom" #Must be unique in Heroku
          heroku_email: "benoit.cppn@gmail.com"
          appdir: "Streamlit" #
```


Mise en ligne

- Utilisation de Heroku pour l'hébergement :
 - API
 - Dashboard interactif

L'API est utilisé pour récupérer les données du client choisi depuis le dashboard puis renvoie un score pour démontrer s'il s'agit d'un bon ou mauvais client

Axe d'amélioration

- améliorer le features engineering
- améliorer les données
- Connaître le coût réel d'un faux négatif pour la fonction métier

Liens externes

<https://github.com/BenCopp/Streamlit>