OpenClassrooms

README.md

Parcours Data Scientist - Projet 7 Implémentez un modèle de scoring

Yann Pham-Van

16/10/2023

Projet7-Implementez_un_modele_de_scoring

Projet d'apprentissage du parcours Data Scientist d'OpenClassrooms en partenariat avec CentraleSupélec.

Mission

- **L'entreprise, avec son offre de crédits, souhaite développer un algorithme de classification indiquant la probabilité de défaut de remboursement.**
- **Un dashboard interactif permet aux chargés de clientèle l'interprétation des prédictions dans un souci de dialogue transparent avec les clients.**
- **Le modèle de scoring, sous forme d'une API, et le dashboard interactif doivent être mis en production séparément.**

Données fournies

Dataset

Résultat des livrables

- **API**
- **Dashboard**

Etapes du projet

1. **Modélisation**

Le notebook Modelisation.ipynb intègre la réutilisation d'un kernel de feature engineering.

Après la prise d'une référence de base à l'aide d'un modèle DummyClassifier, d'autres modèles -linéaires ou non- sont mis en compétition :

- **RandomForestClassifier**
- **LogisticRegression**
- **XGBClassifier**
- **LGBMClassifier**

Une recherche d'hyperparamètres - via GridSearchCV - définit le meilleur d'entre eux en termes d'AUC, de score métier et de temps de calcul. Il s'agit de LGBM.

Il est ensuite optimisé à l'aide d'une recherche plus poussée - via RandomSearchCV et l'outil de tracking des runs d'expérience de MLFlow - des meilleurs hyperparamètres.

Pour optimiser encore plus le modèle et la balance entre les *faux positifs* (manque à gagner) et les *faux négatifs* qui coûtent 10 fois plus chers, le meilleur score métier est recherché en fonction du seuil de décision (*threshold*). Celui-ci est légèrement différent du seuil de 0,5 de base.

2. **API**

Le modèle optimisé est sérialisé afin d'être déployé à l'aide d'une API.

Le code fourni par le fichier app.py est hébergé sur Heroku.

Il peut être testé : ici.

3. **Dashboard**

Le dashboard est fourni par le code de Streamlit/dashboard.py.

Il est aussi hébergé sur Heroku : là.

Il permet de choisir un client depuis la sidebar. Les informations principales du client s'affichent sous le choix de client.

En même temps, les données de ce client sont transmises à l'API qui renvoit la probabilité de défaut.

Dès lors, il est possible d'afficher le résultat de cette probabilité sur une jauge permettant de juger la décision

d'octroi ou non du prêt et aussi l'éloignement relatif au point de bascule, défini par le threshold.

Il est aussi possible, optionnellement à l'aide de cases à cocher, d'afficher :

- **les features les plus déterminantes dans la décision d'acceptation ou refus,**
- **les features les plus importantes à l'échelle de tous les clients,**
- **pour chaque variable, par choix dans une liste déroulante, la situation du client sur 2 histogrammes filtrés sur l'historique des clients acceptés ou refusés.**

4. **Datadrift**

L'analyse du datadrift potentiel est effectué à l'aide de la bibliothèque evidently.

Le code est proposé par le fichier datadrift.ipynb. Le rapport complet est disponible au format html : data_drift_report.html

5. **Packages utilisés**

Listes des environnements nécessaires au bon fonctionnement des notebook et scripts :

- **Modélisation : mlflow_model/requirements.txt**
- **API : requirements.txt**
- **Dashboard : Streamlit/requirements.txt**

6. **Tests unitaires**

L'API voit son code testé, dans le cadre du pipeline de déploiement lors de chaque push sur GitHub, via GitHub Actions : test_app.py