MLOps - TP - MS ESD 2021/2022

L'objectif de ce TP est d'utiliser quelques outils Open Source très utilisés par les Machine Learning Engineers. Seront utilisés entre autres :

- MLFlow
- Black / isort
- Flask
- Pytest
- Git
- Github Actions
- Evidently AI
- Heroku (si le temps vous le permet …)

Le dataset utilisé est disponible à cette adresse : https://archive.ics.uci.edu/ml/datasets/Vertebral+Column . Ce dataset permet de créer un modèle de classification qui prédit la classe d'appartenance (normal, disk hernia or spondilolysthesis) de chaque individu en fonction des caractéristiques de leur colonne vertébrale.

Vous avez à disposition tout le code qui a permis de présenter les différents outils en CM.

1. Dans un notebook, effectuer l'exploratory data analysis afin d'étudier le jeu de données et faire une première modélisation pour prouver que le projet est viable, et que donc il y a un intérêt à aller plus loin et à le déployer en production.
2. Créer un dossier qui contiendra tout votre code puis créer les fichiers requirements.txt et setup.py. Ajouter les packages utilisés au fur et à mesure dans le fichier requirements.txt
3. En suivant les bonnes pratiques d'organisation, scripter votre code afin de créer trois fichiers (Créer un fichier params.yaml avec toutes les variables. Cela permet d'avoir un code propre et de modifier plus rapidement les différents chemins vers les fichiers, la valeur des hyperparamètres etc.) :
    a. load_data.py : qui récupère les données sur UCI et qui sauvegarde un fichier csv servant à entraîner le modèle dans un dossier data/
    b. data_processing.py : qui split vos données en un jeu d'entraînement et un jeu de test + qui effectue un processing si besoin
    c. train.py : qui entraîne un modèle. Utiliser MLFlow pour tracker les performances de votre modèle en faisant varier les hyper paramètres du modèle (faire au moins 2 runs)
4. Créer un fichier production_model_selection.py qui récupère le meilleur modèle dans le registre MLFlow et qui l'enregistre avec joblib dans un dossier models/

5. Documenter le code
6. Formater votre code pour qu'il soit plus lisible avec black et organiser vos imports de la meilleure façon avec isort (il faut bien créer le fichier pyproject.toml comme dans le projet iris)
7. Créer une API Flask afin de pouvoir utiliser le modèle en temps réel :
    a. copier/coller le dossier webapp du projet iris
    b. Changer le model.joblib en le remplaçant avec le vôtre
    c. Dans webapp/templates/index.html modifier/ajouter les balises <textarea> afin que cela corresponde bien avec le modèle.
    d. copier/coller le fichier app.py : il y a une modification à faire dans la valeur de la réponse
8. Écrire 1 ou 2 tests :
    a. Créer le dossier tests/
    b. Créer un fichier test.py et écrire dedans 1 ou 2 tests
    c. Lancer pytest pour voir si vos tests se déroulent correctement
9. Créer un dépôt Git et utiliser Github comme repo distant
10. Mettre en place une pipeline CI/CD avec Github Actions afin de lancer Pytest à chaque push
11. Monitorer les performances avec Evidently AI : pour ce faire, utilisez les mêmes données qu'en entraînement, en considérant que ce sont des nouvelles données. Utilisez le fichier monitoring.py. Ouvrir le fichier résultant "data_and_target_drift_dashboard.html" dans votre navigateur.

Bonus : déploiement sur Heroku afin que le modèle puisse être utilisé sur un serveur web :

1. Aller sur le site https://www.heroku.com
2. Créer un compte
3. Créer un nouveau projet
4. Installer gunicorn (serveur web HTTP WSGI écrit en Python) - pip install gunicorn
5. Copier coller le fichier procfile
6. Copier coller le fichier runtime.txt en remplaçant la version de Python avec celle que vous utilisez dans votre projet
7. lancer les commandes suivantes :
    a. git init
    b. git add .
    c. git commit -m "add files to heroku"
    d. heroku login
    e. heroku git:remote -a <nom_projet_heroku>
    f. git push heroku master

Votre projet devrait maintenant être déployé sur Heroku.
Rendez-vous à cette adresse : https://<nom_projet_heroku>.herokuapp.com/

Rendu final :

- Des captures d’écran de votre serveur MLFlow avec vos runs
- Tout votre dossier contenant tous vos scripts
- Le lien vers votre repo github
- L’adresse web Heroku si vous avez eu le temps.