Documentation Technique – Pipeline ML Dockerisé

Obinna, Flavien ,Haniel

Objectif du Projet

Créer un pipeline complet et reproductible en machine learning permettant de prédire la longueur du sépale d'une fleur à partir de sa largeur, en utilisant le jeu de données Iris. Ce pipeline est conteneurisé avec Docker, suit les bonnes pratiques de séparation des étapes (prétraitement, modélisation, API), et permet de suivre les expérimentations avec MLflow.

Technologies & Outils

Outil / Techno	Rôle dans le pipeline
Docker	Conteneurisation des différents services
PostgreSQL	Stockage structuré des données Iris
MLflow	Suivi des expérimentations et modèles ML
Scikit-learn	Entraînement du modèle de régression
FastAPI	Création de l'API de prédiction
Pandas / psycopg2	Prétraitement et insertion en base

Architecture des Services

```
docker-compose.yml
├── preprocessing     # Charge iris.csv → PostgreSQL
├── modeling          # Entraîne un modèle LinearRegression
├── mlflow            # Interface de suivi des expériences (port 5000)
├── api (FastAPI)     # Expose un endpoint /predict (port 8000)
├── postgres          # Stocke les données (port 5432)
```

Étapes Techniques

1. Prétraitement

- Lecture de iris.csv
- Connexion sécurisée à PostgreSQL
- Création de la table iris_data si elle n'existe pas
- Insertion des données nettoyées

2. Modélisation

- Chargement des données depuis PostgreSQL
- Entraînement d'un modèle de régression linéaire
- Enregistrement des paramètres, métriques et artefacts avec MLflow

3. API de Prédiction

- Chargement du modèle sauvegardé
- Endpoint /predict?sepal_width=...
- Retourne un JSON avec la longueur prédite du sépale

Déploiement Docker

docker compose build
docker compose up

Accès

- Interface MLflow : http://localhost:5000
- API de prédiction : http://localhost:8000/predict?sepal_width=3.2

Problèmes rencontrés & solutions

Problème	Solution apportée
Incompatibilité entre versions de sklearn	Forcer la version compatible dans requirements.txt
PostgreSQL non disponible à l'API	Ajout d'un depends_on et gestion du timeout
Connexion inter-containers	Utilisation des noms de service comme hostname

Arborescence du projet

```
projet-iris-ml/
├── api/
│   └── main.py
├── modeling/
│   └── train_model.py
├── preprocessing/
│   └── preprocessing.py
├── data/
│   └── iris.csv
├── Dockerfile (x3)
├── docker-compose.yml
└── README.md
```

Pistes d'amélioration

- Ajouter des tests unitaires pour le modèle et l'API
- Intégrer un orchestrateur comme Airflow pour automatiser les tâches
- Passer à des modèles plus puissants (RandomForest, XGBoost)
- Mettre en place un workflow CI/CD avec GitHub Actions
- Déploiement cloud (Heroku, Render, AWS, etc.)

Ressources

GitHub du projet : https://github.com/Haniel-svg/Projet-Iris-Data-Pipeline