mlflow un outil de gestion de machime learning, en particulier, on mettra le focus sur une fonctionnalité pariculière :

le mlflow tracking qui permet de gérer et de suivre les différents expérimentations, il

est le composant de journalisation et de gestion des expériences de la plateforme MLflow

et de pouvoir theoriser les résultats au fur et à mesure avec les algos à tester

les paramètres qu'on a fixé et les résultats qu'on a obtenus et cela permet un bon suivi.

et à la fin quand on a testé différents approches.

Il vous permet d'enregistrer et d'interroger les données relatives à vos expériences d'apprentissage automatique,

telles que les métriques(ils sont utilisées pour enregistrer les résultats d'évaluation, tels que la précision, la perte ou le score F1),

les paramètres(ils capturent les hyperparamètres et d'autres détails de configuration pour votre expérience),

et les artefacts(des fichiers générés au cours de votre exécution, tels que des modèles formés,

des visualisations de données ou des journaux).

Avec MLflow Tracking,

vous pouvez assurer la reproductibilité, rationaliser la comparaison des modèles et

centraliser les données d'expérience pour une meilleure collaboration.

Enfin,c'est le composant de journalisation et de gestion des expériences de la plateforme MLflow.

on est soumit aux fluctuations de l'échantillonnage et la variabilité du résultat,

il faut recenser un outil qui permet la recherche du meilleur modèle.

D'autre part, pycaret crée un fichier log :logs.log, la fonction setup(loq_experiment=True) permet le suivi des expérimentations ,

teste un ensemble d'algorithmes en recherchant le plus efficace, on crée une session d'expérimentations et

l'exécution du setup() crée un fichier mlruns qui stocke les résultats.

tous les résultats sont enregistrés dans mlruns, il y a une possibilité d'avoir un suivi global avec mlflow

(suivi de tous les résultats qui ont été générés - WEB)

on peut lancer un serveur web qui permet de consulter plus facilement le contenu de mlruns.

mlflow enregistre le modèle en fichier pickle, il montre comment on peut l'importer en déploiement

pour les données mise à jour, on redémarre le Kernel.

MLflow Tracking est le composant de journalisation et de gestion des expériences de la plateforme MLflow.

Il vous permet d'enregistrer et d'interroger les données relatives à vos expériences d'apprentissage automatique,

telles que les métriques, les paramètres et les artefacts. Avec MLflow Tracking,

vous pouvez assurer la reproductibilité, rationaliser la comparaison des modèles et

centraliser les données d'expérience pour une meilleure collaboration.

Visualisation des expériences dans l'interface MLflow

MLflow fournit une interface web pour visualiser vos expériences et vos exécutions. Lancez-la avec :

mlflow ui

Naviguez jusqu'à http://localhost:5000 pour l'explorer :

Le résultat est un DataFrame pandas, ce qui facilite l'analyse ou la visualisation de vos données

d'expérience de manière programmatique.

Autre composant important c'est le MLflow Model Registry qui est un référentiel centralisé pour tous vos modèles d'apprentissage automatique.

Il fournit un contrôle de version, des transitions d'étapes du cycle de vie et des outils de collaboration,

assurant une approche rationalisée de la gestion des modèles. Que vous déployiez, testiez ou archiviez des modèles,

le registre permet de tout organiser.

Chaque modèle enregistré se voit attribuer une version unique, ce qui facilite le suivi des itérations au fil du temps.

Le registre des modèles suit les modèles par versions et les organise en étapes prédéfinies :

Aucune : Stade par défaut pour les modèles nouvellement enregistrés.

Staging : Modèles en cours de validation ou de test.

Production : Modèles déployés pour une utilisation en production.

Archivé : Modèles obsolètes qui ne sont plus utilisés.

Conclusion

MLflow est un outil puissant pour gérer le cycle de vie de l'apprentissage automatique de bout en bout.

En intégrant ses composants, vous pouvez vous assurer que vos workflows sont organisés, reproductibles et prêts pour la production.

Après ce long rappel nécessaire qui explique la démarche MLops et la solution retenue:

Objectif

Après l'exécution du pipeline mlflow-loan-rm.py → mlflow-loan-om.py → mlflow-loan-register.py, le notebook doit :

Afficher toutes les tables PostgreSQL MLflow (experiments, runs, metrics, model_versions, etc.).

Afficher les modèles et leurs versions enregistrées sous forme de DataFrame.

Le module psycopg2 est ajouté dans l'URI de connexion de mlflow.set_tracking_uri("postgresql+psycopg2://...")

pour spécifier le pilote utilisé pour se connecter à la base de données PostgreSQL.

Explication détaillée :

PostgreSQL en tant que backend store :

MLflow utilise une base de données pour stocker les métadonnées des expériences, des modèles et des artefacts.

PostgreSQL est une base de données relationnelle couramment utilisée pour cela.

SQLAlchemy et dialecte PostgreSQL :

MLflow utilise SQLAlchemy pour interagir avec la base de données.

SQLAlchemy requiert un dialecte et un pilote pour établir une connexion.

postgresql+psycopg2:// signifie que l'on utilise PostgreSQL avec psycopg2 comme pilote pour gérer la connexion.

Rôle de psycopg2 :

psycopg2 est un adaptateur Python pour PostgreSQL.

Il permet aux bibliothèques comme SQLAlchemy (et donc MLflow) d'envoyer des requêtes SQL à PostgreSQL.

Sans psycopg2, MLflow ne pourrait pas établir une connexion avec la base de données PostgreSQL.

Automatiser les expérimentations avec GitHub Actions + MLflow permet d'intégrer un workflow MLOps

efficace et reproductible. Voici les étapes clés :

Objectifs de l'Automatisation

Lancer des expérimentations MLflow automatiquement après chaque commit/pull request.

Enregistrer les métriques et modèles dans MLflow.

Déployer un modèle automatiquement après validation.

Après un push sur main, GitHub Actions exécutera les étapes suivantes dans cet ordre :

✅ mlflow-loan-rm.py → recherche du modèle

✅ mlflow-loan-om.py → optimisation du modèle

✅ mlflow-loan-register.py → enregistrement du modèle dans MLflow

✅ mlflow-results.py → affichage des résultats dans PostgreSQL et MLflow UI

Résumé des améliorations

✅ Le notebook est maintenant intégré au pipeline

✅ Il affiche toutes les informations MLflow sous forme de DataFrame dans Jupyter

✅ GitHub Actions exécute automatiquement le notebook après le pipeline

Modules utilisés dans mlflow-results.py

Les principaux modules utilisés sont :

Module	Utilité
mlflow	Connexion au tracking MLflow
MlflowClient	Gestion des modèles dans le Model Registry
pandas	Chargement et affichage des données
sqlalchemy.create_engine	Connexion à PostgreSQL
ace_tools_open	Affichage des DataFrames dans Jupyter Notebook

Microsoft Windows [version 10.0.19045.5555]

C:\Users\lucie\AppData\Local\Programs\pgAdmin 4\runtime>"C:\Users\lucie\AppData\Local\Programs\pgAdmin 4\runtime\psql.exe"

"host=localhost port=5432 dbname=mlflow_db user=postgres sslmode=prefer connect_timeout=10" 2>>&1

psql (17.2, server 16.8)

WARNING: Console code page (850) differs from Windows code page (1252)

8-bit characters might not work correctly. See psql reference

page "Notes for Windows users" for details.

Type "help" for help.

mlflow_db=# \dt

List of relations

Schema	Name	Type	Owner
public	alembic_version	table	postgres
public	experiment_tags	table	postgres
public	experiments	table	postgres
public	latest_metrics	table	postgres
public	metrics	table	postgres
public	model_version_tags	table	postgres
public	model_versions	table	postgres
public	params	table	postgres
public	registered_model_tags	table	postgres
public	registered_models	table	postgres
public	runs	table	postgres
public	tags	table	postgres

(12 rows)

Après l'exécution du pipeline CI/CD, en particulier, le composant mlflow-results.py

toutes les tables issus de MLruns sont alimentées et présentées dans un DataFrame.