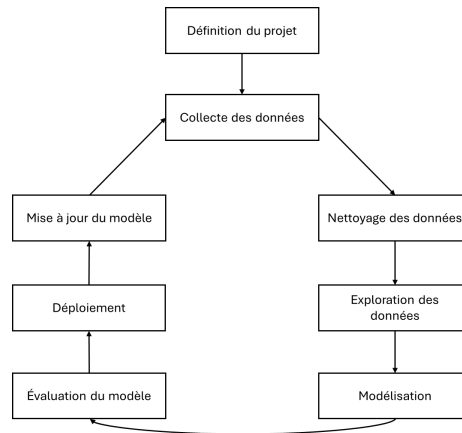


Atelier MLFLOW:

Bienvenue dans ce TD réalisé par Dounya Bourhani, Paule Naomi Kaldjob, Albane Nicoullaud et Fiona Steiner.

Ce TD est consacré à explorer le cycle de vie et l'industrialisation d'un projet de machine learning.



Le schéma ci-dessus illustre le cycle de vie d'un projet de machine learning, qui sera notre feuille de route tout au long de ce TD.

Dans ce TD, vous allez découvrir deux outils :

- La plateforme de machine learning : **MLFlow**
- L'orchestrateur de workflow : **AirFlow**

Documentation:

- Documentation MLFlow : https://mlflow.org/docs/latest/python_api/index.html
- Documentation AirFlow : <https://airflow.apache.org/docs/>

Outil n°1: MLFLOW

1 Définition du projet:

Dans le cadre d'une initiative visant à avancer dans la lutte contre le cancer, un projet ambitieux est lancé pour explorer les possibilités offertes par les données de diagnostic du cancer du sein. Ce projet, d'une importance capitale, vise à développer des modèles prédictifs précis pour distinguer les tumeurs malignes des tumeurs bénignes, s'appuyant sur l'analyse de données issues de recherches cliniques.

2 Collecte des données:

La première étape de cette mission consiste à acquérir un ensemble de données fiable et complet. Les données ciblées pour cette recherche proviennent de la collection "Breast Cancer Wisconsin", un référentiel reconnu pour sa qualité et sa pertinence dans le domaine de l'apprentissage automatique et de la recherche médicale. L'accès à ces données est obtenu après une navigation prudente et sécurisée à travers diverses sources en ligne, culminant dans l'obtention du fichier nommé cancer45.csv.

3 Nettoyage des données:

L'examen attentif de cet ensemble de données révèle une qualité exceptionnelle, avec aucune donnée manquante et aucune anomalie apparente. Cette pureté des données permet une analyse plus fiable et plus précise, éliminant le besoin de procédures de nettoyage complexes ou de corrections de données.

4 Exploration des données :

Cette base de données comprend 450 individus, chaque instance étant caractérisée par 30 attributs numériques dérivés d'images de tissus mammaires. Les données sont étiquetées selon leur diagnostic : "maligne" ou "bénigne", fournissant une base solide pour l'entraînement et l'évaluation de modèles de machine learning.

5 MLFlow : Modélisation - Evaluation des données:

Dans un premier temps, il nous faut installer les outils nécessaires.
Placez-vous dans votre répertoire de travail et **créer un nouvel environnement appelé `mlflow-env` et activez-le pour installer la librairie `mlflow`** et les librairies nécessaires à un projet de machine learning (pandas, seaborn...).

Vérifiez la bonne installation de MLFlow en lançant l'interface utilisateur :
`mlflow server -host 127.0.0.1 -port 5000`.

Dans ce TD, nous allons travailler avec Python, l'idée ici est de générer un grand nombre d'exécutions avec des modèles et hyper-paramètres différents.

Nous allons travailler avec les modèles suivants :

- DecisionTree
- LogisticRegression
- SVC
- GaussianNB
- KNeighbors

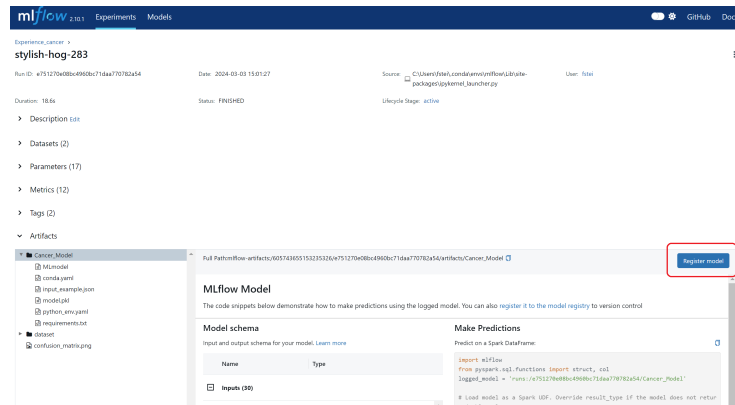
Récupérer le notebook Python nommé : MLFLOW-Experiences-Exercice.ipynb, lisez attentivement toutes les cellules du notebook afin de pouvoir le compléter et lancer l'exécution.

Sur l'interface dans l'onglet Chart, observez les différentes exécutions et leurs métriques et sélectionnez en une première parmi les critères suivants par exemple :

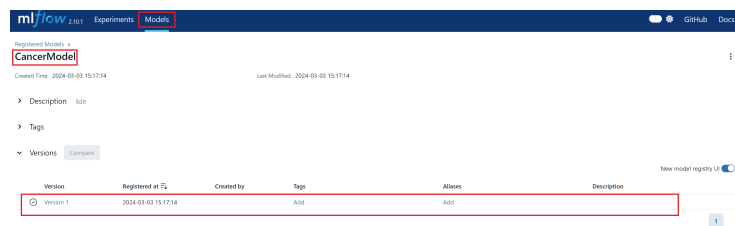
- La meilleure métrique Accuracy/Precision...
- La plus rapide

Enregistrer le modèle avec le bouton Register Model dans un nouvel Register model. Suivez les instructions et nommez ce registry model Cancer_Model.

Atelier 3 : Cycle de vie d'un projet de machine learning



Votre modèle apparaît maintenant dans le Registered Model (Onglet Models).



Ajoutez un Tag et Alias comme sur la capture suivante :

Version	Registered at	Created by	Tags	Aliases
Version 2	2024-03-03 15:35:00		statut: à valider	Add
Version 1	2024-03-03 15:17:14		statut: à mettre en production	@ riccorakotomalala

Faites en de même sur deux ou trois modèles et téléchargez leur model.pkl. Ils nous serviront pour les prédictions.



6 Prédiction et déploiement:

Prédictions:

Récupérez le fichier [MLFLOW-Predictions-Exercice.ipynb](#) et complétez-le.

Déploiement - Production:

Déconnectez-vous du serveur.

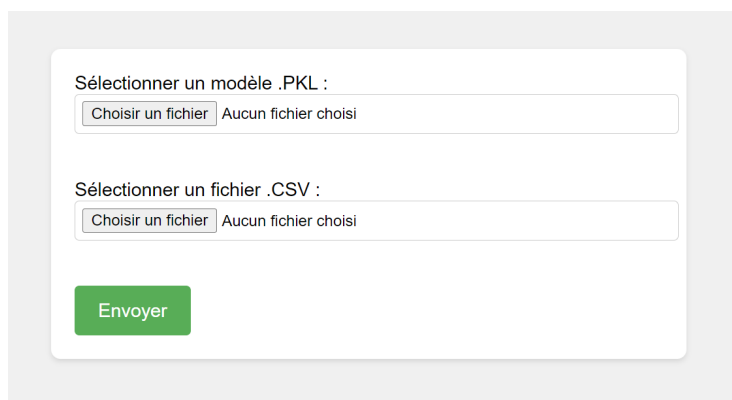
Le déploiement en production sera fait avec un serveur **Flask local**.

Récupérer le dossier [MLFLOW-Production-Exercice](#) qui contient le site Flask.

Dans le fichier `Production/app.py`, ajouter le chemin du répertoire de [MLFLOW-Production-Exercice](#) avec l'instruction `os.chdir(path)`.

Dans une invite de commande, placez-vous dans le dossier `Production` et lancer le fichier `app.py`.

Connectez-vous via votre navigateur à `http://localhost:5000` pour faire vos prédictions avec vos `model.pkl` préalablement télécharger depuis l'interface de MLFlow et les données à prédire. Dans notre cas, `cancer119.csv`.



7 Mise à jour du modèle:

La mise à jour du modèle s'effectue en modifiant le `model.pkl`. Il s'agit, ici, d'une action manuelle, mais dans un réel environnement de production, ceci sera transparent pour l'utilisateur qui n'aurait qu'à insérer ses données d'entrées et visualiser les résultats.

Outil n°2: AirFlow

Prise en main de l'outil:

Afin de vous aider à comprendre l'outil Airflow en douceur, nous vous proposons une petite mise en bouche facile.

L'objectif est que vous puissiez ensuite réaliser un exercice seul.

Installation avec Docker:

1. Exécutez la mise à jour WSL si elle n'est pas déjà faite (`wsl --update`).
2. Lancez l'application Docker.
3. Dans votre bureau, créez un nouveau dossier dédié à *Airflow*. Créez un dossier *file*, contenant *iris110.csv*, dans le dossier Airflow puis lancer le dossier Airflow dans le terminal.
4. Dans le terminal, ouvrez l'éditeur de code.
5. Dans Visual Studio installez l'extension Docker officielle si ce n'est pas déjà fait ainsi que l'extension python.
6. Créer ensuite un nouveau fichier dockerfile et entrez ceci:

```
FROM apache/airflow:latest

USER root

RUN apt-get update && \
    apt-get -y install git && \
    apt-get clean

USER airflow

RUN pip install scikit-learn
COPY /file/iris110.csv /usr/src/app/
```

Cette commande crée une image Docker personnalisée basée sur l'image Docker officielle d'Apache airflow.

Nous vous avons aussi ajouté des commandes qui vont vous aider pour l'exercice suivant.

7. Enregistrez le fichier et faire un clic droit sur le fichier Docker et choisissez "Build Image...".

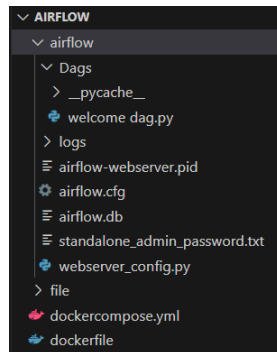
Cette action fournit un nom pour l'image personnalisée dans la barre de recherche en haut. Faites "Entrée" pour télécharger l'image officielle Docker d'Apache Airflow et créer votre propre image personnalisée. Une fois que le message "What's next?" apparaît fermez le terminal.

8. Créez ensuite un fichier dockercompose.yml et écrivez ceci:

```
1  version: '3'
2
3  services:
4    airflow:
5      image: airflow:latest
6      volumes:
7        - ./airflow:/opt/airflow
8      ports:
9        - "8080:8080"
10     command: airflow standalone
11
```

Cette commande ajoute l'image flux d'airflow personnalisée que nous avons créée précédemment. Elle relie le répertoire de flux d'airflow du dossier actuel sur notre ordinateur local au répertoire opt airflow à l'intérieur du conteneur.

9. Enregistrez le fichier, faites un clic droit dessus et choisissez *Compose up*. Fermez ensuite le terminal
10. Retournez sur l'application Docker et vérifiez que votre conteneur apparaît bien.
11. Ouvrez le et sélectionnez *View details*
12. Cliquez sur le lien localhost en haut
13. Dans *Username* mettez Admin et dans *Password* mettez celui qui apparaît ici:



Développer et planifier un Airflow Dag:

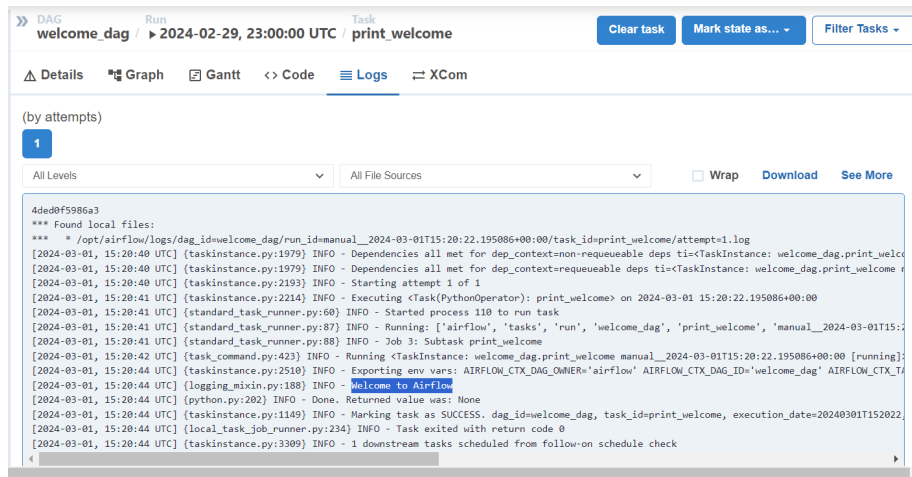
Dans votre dossier *airflow* dans Visual Studio créez un nouveau répertoire appelé *Dags* dans lequel vous déposez le fichier `welcome_dag.py` et complétez le code qui se trouve dans le fichier. Puis enregistrez le.

Retournez sur votre page Airflow, et attendez quelques minutes que "welcome-dag" s'affiche.

Une fois qu'il s'est affiché cliquez dessus. Pour l'exécuter, cliquez sur le bouton "Trigger Dag" en haut à droite.



Dans "Graph" et vérifiez que vos trois tasks aient été exécuté avec succès. Puis, vérifiez les journaux respectifs en allant dans "Logs".



The screenshot shows the Apache Airflow web interface. At the top, there's a navigation bar with 'DAG', 'Run', and 'Task' tabs. The current view is for the 'welcome_dag' DAG, specifically the 'print_welcome' task, with a timestamp of '2024-02-29, 23:00:00 UTC'. There are buttons for 'Clear task', 'Mark state as...', and 'Filter Tasks'. Below the navigation bar, there are tabs for 'Details', 'Graph', 'Gantt', 'Code', 'Logs', and 'XCom'. The 'Logs' tab is selected, showing a list of logs for the task. The logs are filtered by attempt (1) and show all levels. The log content includes information about the task's execution, such as the DAG ID, task ID, and the command used to run the task. The log ends with a message indicating that the task was successful and that downstream tasks are scheduled.

Bravo ! Vous savez maintenant utiliser une partie de l'outil Airflow. Passons maintenant aux choses sérieuses...

Premier modèle de machine learning

Description des données

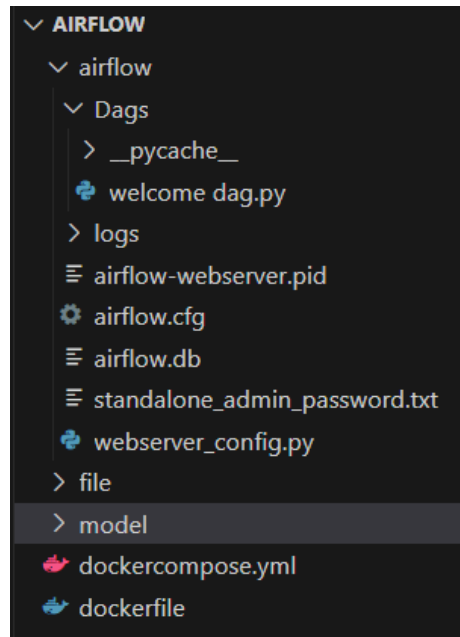
Afin de vous familiariser avec Ariflow, vous allons travailler sur les données iris (que l'on connaît très bien maintenant). Les données iris proviennent de l'étude de trois espèces d'iris : setosa, versicolor et virginica. La base de donnée que l'on vous fournis contient 110 observations. Chacune d'entre elles décrites par : la *longueur* et la *largeur* du *sépale* ainsi que la *longueur* et la *largeur* du *pétale*.

Construction du modèle

L'objectif de cette deuxième partie est de vous construire des modèles de machine Learning:

1. Créez le dossier nommé *model* dans le dossier *airflow* précédemment créé
2. Ajoutez le fichier *iris110.csv* dans le dossier *file*
3. Ajoutez un autre fichier "secondml.py" dans le dossier *Dags*
4. commencez par créer une première fonction *prepare_data()*.

- Chargez le jeu de données iris dans cette fonction (Le chemin du fichier iris est "/usr/src/app/iris110.csv")



- Divisez les données en ensembles d'entraînement et de test.
- Retournez les données d'entraînement (X_{train} et y_{train}) et les données de test (X_{test} et y_{test}) sous forme de liste.

5. Passons à la deuxième fonction `train_model()`

- Récupérez les variables retournées de la fonction précédente à l'aide de la commande : `kwargs['ti'].xcom_pull()`
- Instanciez le modèle de régression logistique
- Entraînez le modèle sur les données d'entraînement
- Prédire sur les données test
- Calculez la précision et le rappel du modèle
- Utilisez la fonction `pickle.dump()` pour enregistrer le modèle
- Stockez la précision du modèle dans un objet XCom utilisant la commande: `kwargs['ti'].xcom_push()`

6. Sortez maintenant de la dernière fonction et suivez ces étapes:

- Créez un objet "default_args" qui définit les arguments par défaut pour le DAG, tels que le propriétaire('airflow'), la date de début (la date d'aujourd'hui) et le nombre de tentatives de reprise en cas d'échec qui sera égale à 1
- Créez un autre objet "dag". Dans cet objet, Spécifiez le nom du DAG("ml_wf_2"), l'argument par défaut (default_args), une description ('A DAG to train and deploy a machine learning model') et un intervalle de planification journalière
- Définissez deux tâches: "prepare_data_task" et "train_model_task" à l'aide de la commande PythonOperator. Dans cette commande, vous devez nommez les tâches ('prepare_data' et 'train_model'), spécifiez les fonctions Python précédemment créé dans la bonne tâche, associez la tâche à l'objet DAG et définissez le contexte d'exécution dans la commande.
- Définissez les dépendances entre les tâches dans Apache Airflow.

7. Allez dans la plateforme AIRFlow. Vous devez apercevoir un deuxième DAG d'ici quelques minutes. Cliquez dessus et cliquez sur le trigger Dag.

8. Si l'exécution du DAG est réussi:

- Allez dans le terminal de Visual Studio Code et tapez la commande:
docker ps.
- Récuperez l'identifiant du conteneur docker tapez la commande :
docker cp identifiant_conteneur_docker:/tmp/model.pkl model/model.pkl
- Récuperez le fichier pickle dans votre dossier *model*

Maintenant que vous avez le fichier .pkl, vous pouvez le tester sur le serveur Flask vu au point 6 de MLFlow.

Exercice bonus pour les plus rapides

Si vous avez terminé les exercices précédents, essayez de lancer une execution R sur MLFlow tracker.

Cette fois, l'exercice n'est pas guidé mais vous pouvez suivre ces quelques indications :

- Indiquez le chemin vers l'exécutable mlflow situé dans le dossier Scripts de votre environnement afin que R sache où trouver MLFlow

```
Sys.setenv(MLFLOW_BIN = "C:chemin/vers/mflow/Scripts/mlflow.exe")
```

- Placez-vous dans votre répertoire de travail avec `setwd()`
- La documentation R se trouve à l'adresse : <https://mlflow.org/docs/latest/R-api.html>