Hướng dẫn đào tạo mô hình dự đoán cơ bản

- Mô hình được đào tạo để đưa ra dự đoạn lượng nước cần bơm, dựa trên 3 đặc trưng cơ bản: Nhiệt độ, độ ẩm môi trường và độ ẩm đất.

- Nếu như đã cài đặt môi trường theo file hướng dẫn, các bạn có thể train tại máy của mình. Hoặc nếu các bạn dùng Google Colab, các bạn sẽ cần pip các thư viện cần thiết cho việc import.

1. Import các thư viện cần thiết

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error
import joblib
```

- **numpy (np)** và **pandas (pd)**: hai thư viện cơ bản để xử lý mảng số và bảng dữ liệu.
- **train_test_split**: hàm để chia dữ liệu thành hai phần, một để huấn luyện (train), một để kiểm thử (test).
- **Pipeline**: cho phép xây dựng chuỗi các bước (ví dụ: chuẩn hóa → học máy) thành một "đường ống" duy nhất.
- **StandardScaler**: chuẩn hóa (scale) dữ liệu – đưa mỗi đặc trưng về phân phối chuẩn (mean = 0, std = 1).
- **XGBRegressor**: mô hình XGBoost cho bài toán hồi quy (dự đoán đầu ra số liên tục).
- **mean_squared_error**: để đánh giá sai số giữa giá trị thật và giá trị dự đoán.
- **joblib**: để lưu và nạp mô hình đã huấn luyện (serialization).

---

2. Đọc dữ liệu từ file CSV

```
df = pd.read_csv('data.csv')    # cột target là 'volume_ml'
```

- pd.read_csv('data.csv') đọc file data.csv vào một DataFrame df.
- DataFrame giống như bảng Excel trong Python, có các hàng (bản ghi) và cột (đặc trưng).

---

3. Chuẩn bị tập đặc trưng X và nhãn y

```
X = df[['temperature', 'humidity_env', 'soil_moisture']]
y = df['volume_ml']
```

- **X**: ma trận đầu vào, gồm 3 cột:
    1. temperature (nhiệt độ)
    2. humidity_env (độ ẩm không khí)
    3. soil_moisture (độ ẩm đất)
- **y**: vector đích (target) là cột volume_ml (lượng nước cần bơm, ml) mà chúng ta muốn dự đoán.

---

4. Chia dữ liệu thành tập huấn luyện và kiểm thử

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42
)
```

- test_size=0.2: giữ lại 20% dữ liệu cho kiểm thử, 80% dùng để huấn luyện.
- random_state=42: cố định "hạt giống" ngẫu nhiên để mỗi lần chạy kết quả chia dữ liệu giống nhau (tái lập được).

Kết quả:

- **X_train, y_train**: dùng để huấn luyện mô hình.
- **X_test, y_test**: dùng để đánh giá mô hình sau khi huấn luyện.

---

5. Lưu tập huấn luyện và kiểm thử ra file CSV (giúp kiểm tra dữ liệu)

```
df_train = X_train.copy()
df_train['volume_ml'] = y_train
df_test  = X_test.copy()
df_test['volume_ml']  = y_test

df_train.to_csv('train.csv', index=False)
df_test.to_csv('test.csv',  index=False)
print("Đã lưu train.csv và test.csv")
```

- Tạo 2 DataFrame: df_train và df_test bằng cách ghép X với y tương ứng.
- Ghi ra file train.csv và test.csv (không có cột chỉ số index), để:
    - Dễ theo dõi, in ra, hoặc dùng cho các bước khác.
    - Đảm bảo bạn biết chính xác dữ liệu nào đã dùng để huấn luyện và kiểm thử.

---

6. Định nghĩa pipeline: chuẩn hóa → XGBoost

```
reg_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('xgb', XGBRegressor(
        objective='reg:squarederror',
        n_estimators=200,
        max_depth=6,
        learning_rate=0.1,
        random_state=42
    ))
])
```

- **Pipeline** gồm hai bước:
    1. **scaler**: StandardScaler() sẽ tính mean và std trên tập huấn luyện, rồi chuẩn hóa từng đặc trưng.
    2. **xgb**: XGBRegressor là mô hình XGBoost với các tham số:
        - objective='reg:squarederror': bài toán hồi quy, tối thiểu hóa sai số bình phương.

- n_estimators=200: số lượng cây quyết định (trees).
- max_depth=6: độ sâu tối đa của mỗi cây.
- learning_rate=0.1: tốc độ học (learning rate).
- random_state=42: để kết quả nhất quán.

Pipeline giúp:

- Đảm bảo mỗi lần fit đều áp dụng chuẩn hóa rồi mới huấn luyện.
- Khi predict, pipeline cũng tự động chuẩn hóa dữ liệu mới trước khi dự đoán.

---

7. Huấn luyện mô hình

```
reg_pipeline.fit(X_train, y_train)
```

- .fit(...) thực hiện hai bước:
  1. **fit scaler** trên X_train (tính mean, std).
  2. **huấn luyện XGBRegressor** với dữ liệu đã chuẩn hóa.

Kết quả: pipeline đầy đủ đã "học" mối quan hệ giữa (temperature, humidity_env, soil_moisture) và volume_ml.

---

8. Đánh giá hiệu năng trên tập kiểm thử

```
preds = reg_pipeline.predict(X_test)
preds = np.clip(preds, 0, 50)
rmse = np.sqrt(mean_squared_error(y_test, preds))
print(f'>> Test RMSE: {rmse:.3f} ml')
```

- **Dự đoán** (.predict) cho X_test.
- **np.clip(preds, 0, 50)**: giới hạn kết quả dự đoán trong khoảng [0, 50] ml (theo thực tế bơm).
- **mean_squared_error(y_test, preds)**: tính sai số bình phương trung bình.
- **RMSE** (Root Mean Squared Error): căn bậc hai của MSE, cho độ lệch trung bình giữa dự đoán và thực tế, đơn vị ml.

Ví dụ, RMSE = 3.200 ml nghĩa là trung bình mô hình sai khoảng ±3.2 ml.

9. Lưu pipeline đã huấn luyện để deploy

```
joblib.dump(reg_pipeline, 'xgb_reg_pipeline.joblib')
print("Saved pipeline to xgb_reg_pipeline.joblib")
```

- **joblib.dump**: ghi file nhị phân xgb_reg_pipeline.joblib.
- Khi deploy (đưa mô hình lên server, tích hợp vào ứng dụng), chỉ cần joblib.load('xgb_reg_pipeline.joblib') để nạp lại pipeline với tất cả bước chuẩn hóa và mô hình đã huấn luyện.
- Mô hình này sẽ được triển khai trên Flask