

# Giới thiệu Metaflow, Evidently AI và Keras

## I. Metaflow - Orchestrating ML Workflows Made Simple

**Metaflow** là một framework mạnh mẽ được phát triển bởi Netflix, giúp đơn giản hóa quá trình xây dựng và quản lý pipeline machine learning. Với cú pháp thân thiện, khả năng tích hợp tốt với AWS, Kubernetes, và versioning rõ ràng cho cả dữ liệu và mô hình, Metaflow là công cụ lý tưởng để tổ chức, theo dõi và tái sử dụng các bước trong quy trình ML một cách dễ dàng và có hệ thống.

Trong các project của cuốn sách này, **Metaflow sẽ đóng vai trò là xương sống của hệ thống ML pipeline**, giúp bạn cấu trúc các bước như thu thập dữ liệu, huấn luyện mô hình, đánh giá và triển khai — tất cả trong một luồng rõ ràng và có thể mở rộng.

### Metaflow giúp được gì

- **Thiết kế pipeline dễ dàng:** Viết code ML theo cách tự nhiên nhất bằng Python.
- **Quản lý trạng thái (state):** Tự động lưu trữ đầu ra của từng bước, giúp bạn dễ dàng resume hoặc debug.
- **Phiên bản hóa dữ liệu & mô hình:** Hỗ trợ versioning đầu vào, đầu ra, mô hình, và code.
- **Tích hợp cloud dễ dàng:** Hỗ trợ AWS (S3, Step Functions, Batch...) và có thể mở rộng cho các nền tảng khác.
- **Chạy song song & phân tán:** Dễ dàng chạy các bước tốn tài nguyên theo kiểu phân tán hoặc song song.

### Cách hoạt động

Bạn định nghĩa các bước trong pipeline bằng cú pháp Python rất thân thiện. Ví dụ:

```
from metaflow import FlowSpec, step

class HelloFlow(FlowSpec):

    @step
    def start(self):
        print("Hello from Metaflow!")
```

```
self.next(self.end)

@step
def end(self):
    print("Flow finished.")

if __name__ == '__main__':
    HelloFlow()
```

Chạy bằng dòng lệnh:

```
python hello_flow.py run
```

Tính năng nổi bật

Tính năng	Mô tả ngắn
@step decorator	Xác định các bước trong pipeline.
FlowSpec	Khai báo cấu trúc của flow.
self.next()	Xác định bước tiếp theo trong luồng xử lý.
metaflow run	Chạy pipeline.
metaflow resume	Tiếp tục pipeline từ bước bị lỗi.
Tích hợp MLFlow, Argo	Có thể tích hợp thêm các hệ thống orchestration.

II. Evidently AI

**Evidently AI** là một thư viện mã nguồn mở giúp theo dõi hiệu suất của mô hình machine learning trong môi trường sản xuất. Nó cung cấp các báo cáo trực quan về drift dữ liệu, độ lệch phân phối đầu vào/đầu ra, độ chính xác, và nhiều chỉ số quan trọng khác mà bạn cần để đảm bảo mô hình hoạt động ổn định theo thời gian.

Trong các project của sách, **Evidently AI sẽ được sử dụng để thiết lập các bước kiểm tra và giám sát mô hình**, giúp bạn nhanh chóng phát hiện và xử lý các vấn đề như concept drift, data quality issues, và sai lệch hiệu suất trong môi trường thực tế.

Được thiết kế để hoạt động **trước, trong, và sau khi triển khai** mô hình ML.

Evidently AI làm được gì?

Tính năng	Mô tả
Data Drift Detection	Phát hiện khi dữ liệu đầu vào thay đổi so với training data.
Model Performance Monitoring	Theo dõi accuracy, precision, recall,... theo thời gian.

Tính năng	Mô tả
Data Quality Checks	Kiểm tra sự thiếu hụt, phân bố, outliers,... trong dữ liệu.
Tạo báo cáo HTML trực quan	Tạo dashboard HTML dễ hiểu, chia sẻ dễ dàng cho đội ngũ.
Tích hợp dễ dàng	Chạy tốt cùng với Pandas, Jupyter, Airflow, MLflow, v.v.

## Một số use-case phổ biến

- Kiểm tra dữ liệu đầu vào có giống training data không.
- Kiểm tra mô hình có bị **concept drift** không.
- Theo dõi mô hình trong môi trường sản xuất.
- Tạo báo cáo ML định kỳ cho business/stakeholder.

## Ví dụ dùng trong Jupyter Notebook

```
import pandas as pd
from evidently.report import Report
from evidently.metric_preset import DataDriftPreset

# Giả sử bạn có training data và current data
train_df = pd.read_csv("train.csv")
current_df = pd.read_csv("current.csv")

report = Report(metrics=[
    DataDriftPreset()
])

report.run(reference_data=train_df, current_data=current_df)
report.show(mode="inline") # hiển thị trong Jupyter
report.save_html("data_drift_report.html") # xuất báo cáo
```

## Cài đặt

```
pip install evidently
```

## Evidently rất hữu ích nếu bạn là...

- **Data Scientist** cần đảm bảo mô hình hoạt động tốt ngoài thực tế.
- **MLOps Engineer** cần hệ thống giám sát tự động hóa.
- **Business Analyst** muốn hiểu mô hình bằng các biểu đồ dễ hiểu.

## Tích hợp với các công cụ khác

- **Jupyter Notebook** – để khám phá dữ liệu.
- **Airflow** – để lập lịch giám sát định kỳ.
- **MLflow, Metaflow, Dagster** – để theo dõi toàn bộ pipeline.
- **Grafana / Prometheus** – để push metric production theo thời gian thực (với Evidently Service).

## II. Keras và Backend

- **Keras** là một thư viện **high-level API** (giao diện cấp cao) được sử dụng để xây dựng và huấn luyện các mô hình học sâu (deep learning). Keras được thiết kế để dễ sử dụng và đơn giản hóa quy trình phát triển mô hình.
- Keras có thể **hoạt động trên nhiều backend khác nhau** (phần cốt lõi chịu trách nhiệm tính toán số học và tối ưu hóa mô hình), và hiện tại hỗ trợ ba backend chính:
  - **TensorFlow** (là backend mặc định)
  - **Theano** (được hỗ trợ trước đây nhưng đã ngừng phát triển)
  - **CNTK** (Microsoft Cognitive Toolkit)
  - **JAX** (từ Google)

### Vì sao chọn JAX làm backend?

**JAX** là một thư viện từ Google cho **tính toán khoa học**, nổi bật với các tính năng:

1. **Tính toán tự động đạo hàm** (autograd).
2. **Tính toán song song và phân tán** với GPU/TPU.
3. **Vectorization** – thực hiện tính toán trên nhiều phần tử cùng lúc.
4. **Hỗ trợ tối ưu hóa và gradient-based learning** cực kỳ mạnh mẽ.

Khi bạn đặt `KERAS_BACKEND=jax`, bạn yêu cầu Keras sử dụng JAX để thực hiện các phép toán, thay vì TensorFlow

### Cách thức hoạt động với JAX

Khi sử dụng JAX làm backend cho Keras, bạn sẽ được hưởng các ưu điểm nổi bật của JAX, như:

- **Hiệu suất** cao hơn, đặc biệt với việc chạy trên **TPU/GPU**.
- **Tự động đạo hàm (autograd)** mạnh mẽ, cho phép tính toán gradient một cách hiệu quả và linh hoạt.
- **Tối ưu hóa gradient** cho các mô hình học sâu, rất hữu ích khi huấn luyện các mô hình phức tạp.

### Cài đặt JAX và Keras với JAX backend

Để sử dụng **JAX** làm backend cho Keras, bạn cần cài đặt các thư viện cần thiết:

1. Cài đặt Keras Core và JAX:

```
pip install keras-core jax jaxlib
```

2. Sau đó, đặt biến môi trường `KERAS_BACKEND` thành `jax`:

```
export KERAS_BACKEND=jax
```

Bây giờ Keras sẽ sử dụng JAX làm backend khi bạn xây dựng và huấn luyện mô hình.

Ví dụ

Ví dụ, nếu bạn muốn sử dụng Keras với JAX làm backend, bạn có thể tạo mô hình như sau:

```
from keras import layers, models
from keras.datasets import mnist

# Tải dữ liệu MNIST
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Chuẩn bị dữ liệu
x_train, x_test = x_train / 255.0, x_test / 255.0

# Xây dựng mô hình
model = models.Sequential([
    layers.Flatten(input_shape=(28, 28)),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(10)
])

# Biên dịch và huấn luyện
model.compile(optimizer='adam', loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)
```

**Tác giả: Đỗ Ngọc Tú**  
**Công Ty Phần Mềm VHTSoft**

Phiên bản #1

Được tạo 19 tháng 4 2025 13:54:18 bởi Đỗ Ngọc Tú

Được cập nhật 19 tháng 4 2025 14:13:16 bởi Đỗ Ngọc Tú