**Mục tiêu:** Xây dựng một hệ thống có khả năng dự đoán nguy cơ mắc bệnh tim dựa trên dữ liệu đầu vào của người dùng, đồng thời quản lý vòng đời mô hình một cách có hệ thống bằng MLflow.

**Workflow Tổng thể:**

Workflow của dự án có thể chia thành hai giai đoạn chính:

1. **Giai đoạn Huấn luyện & Quản lý Mô hình (Offline - Chạy train.py)**
2. **Giai đoạn Dự đoán & Triển khai (Online - Chạy app.py)**

**1. Giai đoạn Huấn luyện & Quản lý Mô hình (train.py)**

Giai đoạn này tập trung vào việc chuẩn bị dữ liệu, xây dựng, đánh giá và lưu trữ mô hình tốt nhất.

* **Bước 1: Tải và Chuẩn bị Dữ liệu Gốc:**
  + Script đọc dữ liệu từ file heart.csv.
  + Xác định cột mục tiêu (num) và chuyển đổi nó thành dạng nhị phân (0: không bệnh, 1: có bệnh) lưu vào cột target.
  + Loại bỏ các cột không cần thiết cho việc huấn luyện (ví dụ: id, dataset).
  + Tách dữ liệu thành tập đặc trưng (X - chứa các chỉ số sức khỏe *gốc*) và tập nhãn (y - chứa cột target nhị phân).
  + **Lưu lại tên các cột đặc trưng gốc** vào file feature\_names.txt. (Bước này quan trọng cho ứng dụng web sau này).
* **Bước 2: Định nghĩa Quy trình Tiền xử lý:**
  + Xác định các cột nào là số (numerical) và cột nào là phân loại (categorical) trong tập đặc trưng X *gốc*.
  + Định nghĩa các bước xử lý riêng cho từng loại:
    - **Số:** Điền giá trị thiếu (NaN) bằng giá trị trung vị (median), sau đó chuẩn hóa dữ liệu bằng StandardScaler.
    - **Phân loại:** Điền giá trị thiếu (NaN) bằng giá trị xuất hiện nhiều nhất (mode), sau đó mã hóa bằng OneHotEncoder (chuyển thành các cột 0/1).
  + Sử dụng ColumnTransformer để đóng gói các bước xử lý này, đảm bảo mỗi loại cột được xử lý đúng cách. Kết quả là một preprocessor object.
* **Bước 3: Chia Dữ liệu Huấn luyện/Kiểm tra:**
  + Chia tập dữ liệu X và y *gốc* thành tập huấn luyện (X\_train, y\_train) và tập kiểm tra (X\_test, y\_test) để đánh giá hiệu năng mô hình một cách khách quan.
* **Bước 4: Khởi tạo và Thiết lập Thử nghiệm MLflow:**
  + Đặt tên cho thử nghiệm (experiment) trong MLflow (ví dụ: "Dự đoán Bệnh Tim Mạch V2"). Mỗi lần chạy train.py sẽ ghi log vào thử nghiệm này.
* **Bước 5: Huấn luyện, Tinh chỉnh và Ghi log Mô hình:**
  + Lặp qua các loại mô hình muốn thử (ví dụ: Logistic Regression, Random Forest).
  + Với mỗi mô hình:
    - **Tạo Pipeline Hoàn chỉnh:** Kết hợp preprocessor (đã định nghĩa ở Bước 2) với mô hình phân loại hiện tại (base\_classifier) thành một Pipeline hoàn chỉnh. Pipeline này đại diện cho toàn bộ quy trình từ dữ liệu thô đến dự đoán.
    - Lặp qua các tổ hợp siêu tham số (hyperparameters) muốn thử cho mô hình đó (ví dụ: các giá trị C cho Logistic Regression, số cây/độ sâu cho Random Forest).
    - **Bắt đầu một MLflow Run:** Với mỗi tổ hợp tham số, bắt đầu một lần chạy (mlflow.start\_run).
    - **Huấn luyện Pipeline:** fit *toàn bộ pipeline* trên tập huấn luyện (X\_train, y\_train). Pipeline sẽ tự động áp dụng các bước tiền xử lý bên trong nó trước khi huấn luyện mô hình phân loại.
    - **Đánh giá Pipeline:** Sử dụng pipeline đã huấn luyện để dự đoán trên tập kiểm tra (X\_test). Tính toán các chỉ số đánh giá (Accuracy, AUC, F1-Score).
    - **Ghi log vào MLflow Run:**
      * Log các siêu tham số đã sử dụng (mlflow.log\_param).
      * Log các chỉ số đánh giá (mlflow.log\_metric).
      * **Log toàn bộ pipeline đã huấn luyện** (mlflow.sklearn.log\_model). Đây là artifact quan trọng nhất, chứa cả preprocessor đã được fit và mô hình đã được huấn luyện.
    - **So sánh và Lưu trữ Mô hình Tốt nhất:** So sánh chỉ số AUC của run hiện tại với AUC tốt nhất đã ghi nhận. Nếu tốt hơn, lưu lại run\_id và đường dẫn artifact của pipeline này.
* **Bước 6: Đăng ký Mô hình Tốt nhất:**
  + Sau khi thử nghiệm tất cả các mô hình và tham số, xác định pipeline tốt nhất (dựa trên run\_id có AUC cao nhất).
  + Sử dụng mlflow.register\_model để đăng ký artifact pipeline tốt nhất vào **MLflow Model Registry** với một tên cụ thể (ví dụ: QuyTrinhDuDoanBenhTim). MLflow sẽ tự động tạo một phiên bản (Version) mới cho mô hình này.
* **Bước 7: Chuyển Giai đoạn (Staging):**
  + Tự động chuyển phiên bản mô hình vừa đăng ký sang giai đoạn (Stage) "Staging" trong Model Registry. Điều này đánh dấu phiên bản này sẵn sàng cho việc kiểm thử hoặc triển khai bởi ứng dụng web.

**2. Giai đoạn Dự đoán & Triển khai (app.py)**

Giai đoạn này tập trung vào việc sử dụng mô hình đã được quản lý để phục vụ dự đoán cho người dùng cuối thông qua giao diện web.

* **Bước 1: Khởi động Ứng dụng Web:**
  + Khi chạy python app.py:
    - Đọc danh sách tên các đặc trưng *gốc* từ file feature\_names.txt (đã tạo bởi train.py).
    - Kết nối tới MLflow.
    - **Tải Pipeline từ Model Registry:** Sử dụng URI (models:/QuyTrinhDuDoanBenhTim/Staging) để yêu cầu MLflow tải phiên bản pipeline mới nhất đang ở giai đoạn "Staging". loaded\_model bây giờ là đối tượng pipeline hoàn chỉnh đã được huấn luyện.
    - (Lấy thông tin phiên bản đang sử dụng để hiển thị).
* **Bước 2: Hiển thị Giao diện Người dùng (GET Request):**
  + Người dùng truy cập URL của ứng dụng web.
  + Flask render file templates/index.html, hiển thị form nhập liệu với các trường tương ứng với các đặc trưng sức khỏe *gốc*.
* **Bước 3: Nhận và Xử lý Dữ liệu Đầu vào (POST Request):**
  + Người dùng điền thông tin vào form và nhấn nút "Dự đoán Nguy cơ".
  + Trình duyệt gửi dữ liệu form đến server Flask.
  + app.py nhận dữ liệu.
  + **Tạo DataFrame Đầu vào:** Tạo một DataFrame Pandas chỉ với **một hàng** chứa dữ liệu người dùng vừa nhập. **Quan trọng:** Các cột của DataFrame này phải khớp chính xác với feature\_names.txt (tên các đặc trưng *gốc*). Kiểu dữ liệu cũng được chuyển đổi cơ bản (ví dụ age thành int, trestbps thành float, sex giữ nguyên là string).
* **Bước 4: Thực hiện Dự đoán:**
  + Gọi phương thức loaded\_model.predict() và truyền vào DataFrame một hàng vừa tạo.
  + **Bên trong Pipeline (loaded\_model):**
    - Bước preprocessor của pipeline sẽ tự động áp dụng các phép biến đổi (imputation, scaling, one-hot encoding) đã học được từ dữ liệu huấn luyện lên hàng dữ liệu đầu vào này.
    - Dữ liệu đã được tiền xử lý sau đó được chuyển đến bước classifier (ví dụ: Logistic Regression hoặc Random Forest đã được huấn luyện).
    - Classifier đưa ra dự đoán cuối cùng (0 hoặc 1).
  + app.py nhận lại kết quả dự đoán (0 hoặc 1).
* **Bước 5: Hiển thị Kết quả:**
  + Flask render lại file templates/index.html, lần này truyền thêm kết quả dự đoán.
  + Giao diện hiển thị thông báo "NGUY CƠ CAO" hoặc "NGUY CƠ THẤP" dựa trên kết quả, cùng với thông tin về phiên bản mô hình đang được sử dụng.

**Tóm tắt Workflow này :**

* **Tự động hóa:** Quá trình huấn luyện, đánh giá, chọn và đăng ký mô hình được tự động hóa phần lớn.
* **Tái lập:** MLflow lưu trữ mọi thứ (code, data version (nếu tích hợp), tham số, metrics, model), giúp dễ dàng tái lập kết quả.
* **So sánh:** Dễ dàng so sánh hiệu năng của các mô hình/tham số khác nhau qua MLflow UI.
* **Quản lý Mô hình Tập trung:** Model Registry cung cấp nơi quản lý các phiên bản mô hình và các giai đoạn của chúng (Staging, Production).
* **Triển khai Nhất quán:** Ứng dụng web luôn tải mô hình từ một nguồn đáng tin cậy (Model Registry) và sử dụng đúng phiên bản đã được phê duyệt (qua Stage). Pipeline đảm bảo tiền xử lý được áp dụng nhất quán cả khi huấn luyện và dự đoán.