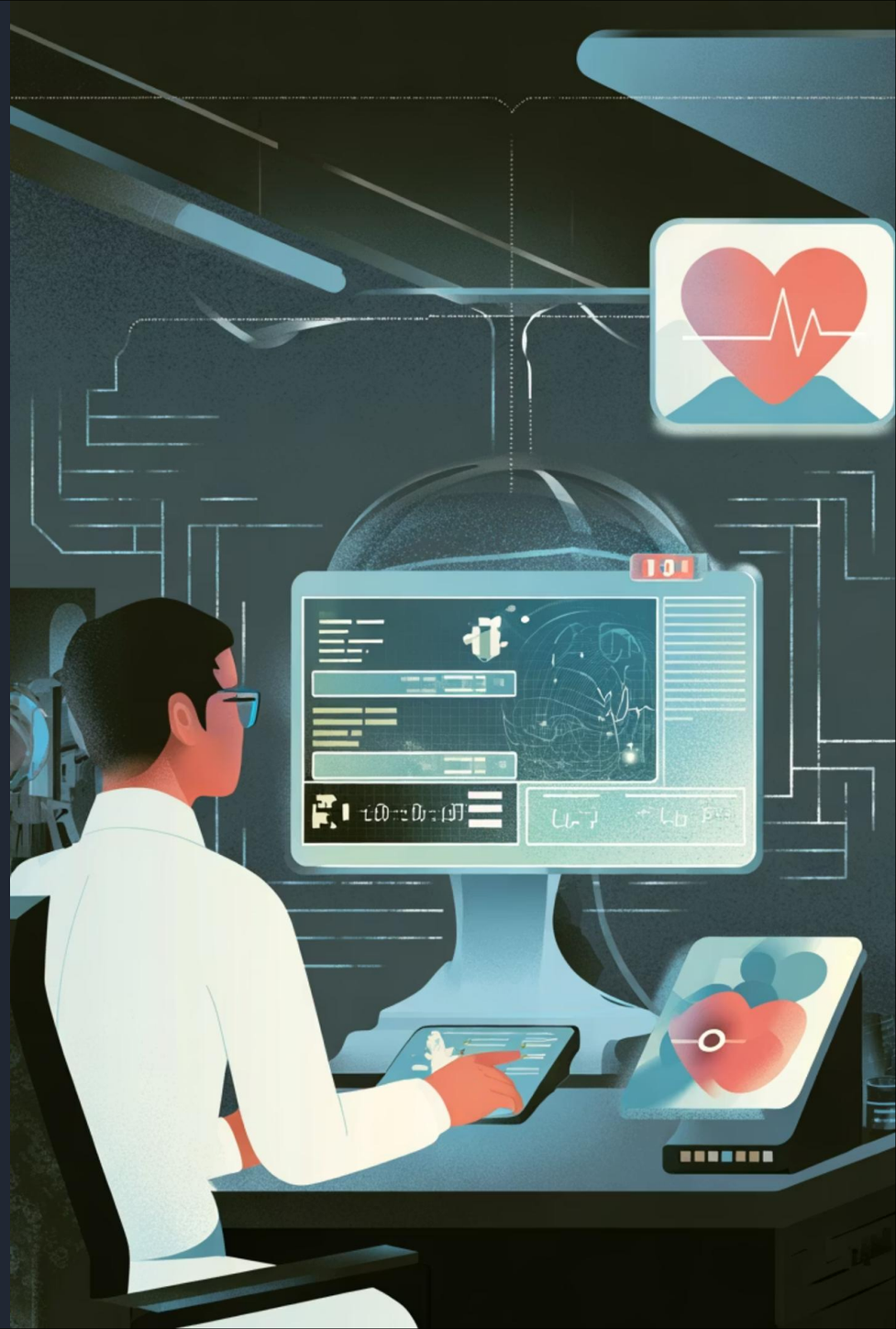
Heart Failure Prediction Pipeline với Spark ML, Kafka và Airflow

Presenter: Lê Văn Hiệp
Student ID: 2102116



Bài toán & Bộ dữ liệu

Bài toán

Dự đoán biến mục tiêu **DEATH_EVENT** (0 hoặc 1) cho bệnh nhân suy tim, nơi 0 là không tử vong và 1 là tử vong. Đây là một bài toán phân loại nhị phân quan trọng trong lĩnh vực chăm sóc sức khỏe.

Bộ dữ liệu

Sử dụng bộ dữ liệu **Heart Failure Clinical Records**, bao gồm **299 bản ghi** và **12 thuộc tính (features)** liên quan đến sức khỏe bệnh nhân.

Một số thuộc tính chính:

- **Ejection Fraction:** Phân suất tống máu, một chỉ số quan trọng về chức năng tim.
- **Age:** Tuổi của bệnh nhân.
- **Serum Creatinine:** Mức creatinine trong huyết thanh, chỉ báo chức năng thận.

Công nghệ sử dụng

Apache Spark ML

Được sử dụng để xử lý dữ liệu lớn, xây dựng các pipeline Machine Learning, huấn luyện và triển khai mô hình một cách hiệu quả.

Apache Kafka

Đóng vai trò là message broker, quản lý luồng dữ liệu streaming theo thời gian thực, đảm bảo việc truyền tải dữ liệu nhanh chóng và đáng tin cậy.

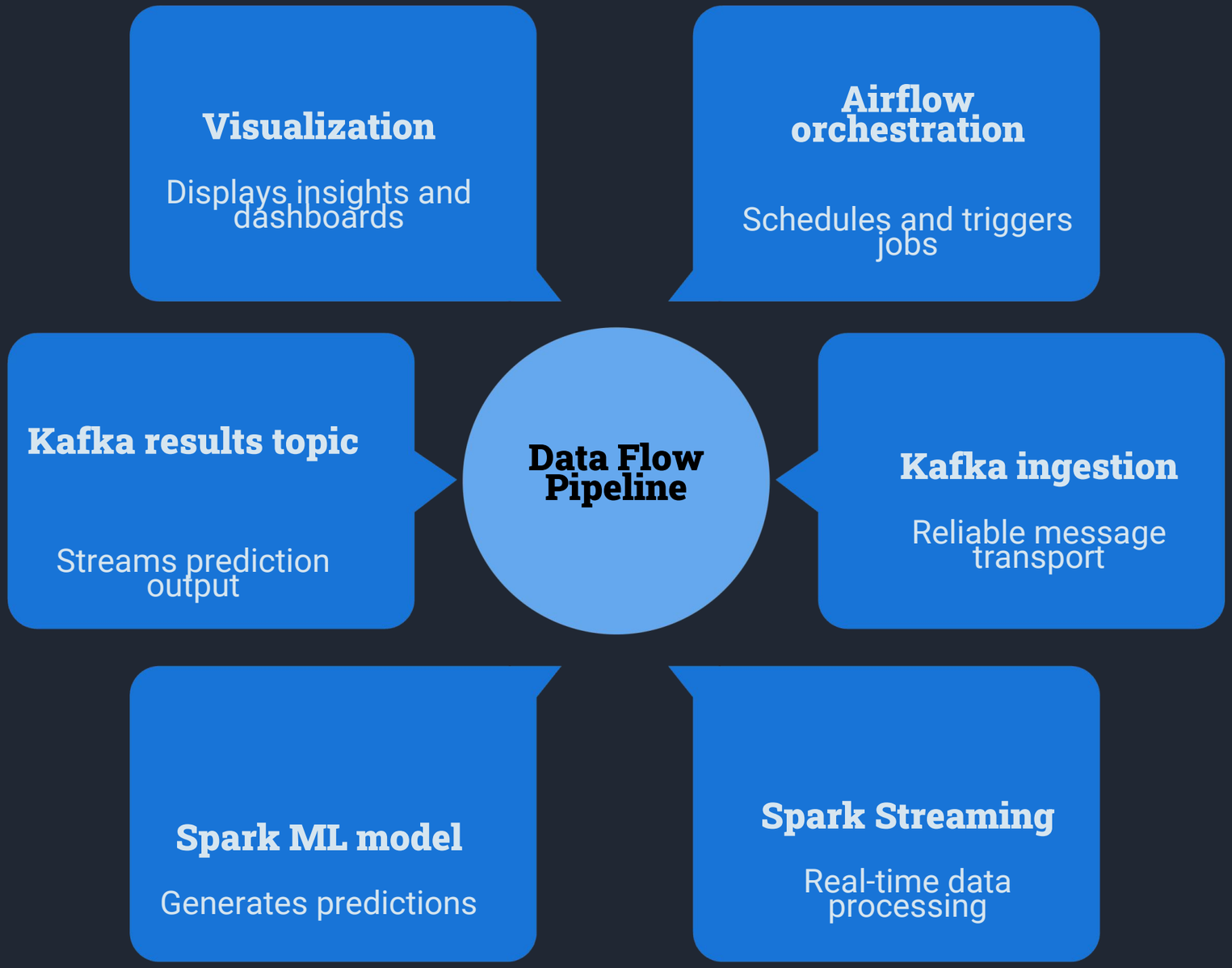
Apache Airflow

Dùng để điều phối toàn bộ workflow của pipeline, tự động hóa các tác vụ và cung cấp khả năng giám sát mạnh mẽ cho từng bước.

Python & Docker

Toàn bộ hệ thống được phát triển bằng Python, và Docker được sử dụng để đóng gói các dịch vụ như Kafka, giúp việc triển khai và quản lý trở nên dễ dàng.

Kiến trúc tổng thể của hệ thống



Quy trình Offline: Xử lý dữ liệu & Huấn luyện mô hình

data_split.py

Script này có nhiệm vụ chia bộ dữ liệu ban đầu thành hai phần: 70% dùng để huấn luyện mô hình và 30% dùng để mô phỏng dữ liệu streaming. Quá trình chia dữ liệu được thực hiện bằng phương pháp stratified split, đảm bảo phân bố nhãn (DEATH_EVENT) được giữ nguyên ở cả hai tập, giúp mô hình được huấn luyện và đánh giá một cách công bằng.

spark_training.py

Sử dụng Apache Spark ML, script này xây dựng một pipeline ML với Random Forest Classifier. Mô hình được đánh giá dựa trên chỉ số AUC Score, đạt khoảng 0.85–0.90, cho thấy khả năng dự đoán tốt. Sau khi huấn luyện, mô hình được lưu lại dưới dạng PipelineModel vào thư mục models/heart_failure_model để tái sử dụng trong giai đoạn streaming.

Quy trình Online: Streaming với Kafka & Spark Streaming

Kafka Topics

heart_failure_input: Topic này tiếp nhận dữ liệu bệnh nhân mới, sẵn sàng cho việc xử lý real-time.

heart_failure_predictions: Sau khi xử lý và dự đoán, kết quả sẽ được ghi vào topic này để các thành phần khác tiêu thụ.

spark_streaming.py

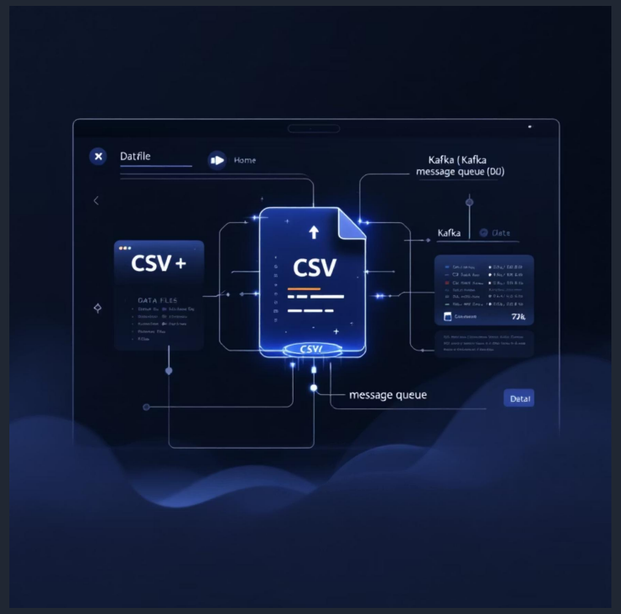
Liên tục đọc dữ liệu từ topic **heart_failure_input**.

- Phân tích dữ liệu JSON thành các cột feature.
- Tải mô hình Spark ML đã được huấn luyện.

Thực hiện dự đoán và ghi kết quả vào topic **heart_failure_predictions**.

Trong giai đoạn online, dữ liệu bệnh nhân mới được đẩy vào Kafka dưới dạng streaming. Spark Streaming job sẽ đọc dữ liệu này, áp dụng mô hình đã huấn luyện để đưa ra dự đoán về nguy cơ tử vong, sau đó kết quả được gửi trở lại Kafka để phục vụ cho các ứng dụng trực quan hóa hoặc các hệ thống khác.

Mô phỏng luồng dữ liệu & Trực quan hóa



stream_simulator.py

Để kiểm tra và minh họa hệ thống, script này đóng vai trò là nguồn dữ liệu real-time. Nó đọc dữ liệu từ file `stream_data.csv` và gửi từng dòng dữ liệu vào Kafka topic **heart_failure_input** với một khoảng thời gian trễ nhất định, mô phỏng luồng dữ liệu y tế thực tế.

visualization_consumer.py

Script này liên tục đọc kết quả dự đoán từ Kafka topic heart_failure_predictions. Dữ liệu nhận được sẽ được dùng để vẽ các biểu đồ real-time, bao gồm:
Predicted vs Actual: So sánh số lượng ca tử vong được dự đoán và thực tế.
Probability over time: Biểu đồ thể hiện xác suất tử vong của bệnh nhân theo thời gian.

Airflow DAG & Điều phối

Apache Airflow đóng vai trò trung tâm trong việc điều phối toàn bộ quy trình dự đoán suy tim, đảm bảo các tác vụ được thực hiện một cách tự động và có thể quản lý được.

01

Khởi động Kafka

Đảm bảo Kafka sẵn sàng nhận và gửi dữ liệu.

02

Tạo Kafka Topics

Thiết lập các topic đầu vào và đầu ra cần thiết.

03

Chia dữ liệu

Thực hiện bước tiền xử lý để tách dữ liệu huấn luyện và streaming.

04

Huấn luyện mô hình

Chạy script Spark ML để huấn luyện và lưu mô hình dự đoán.

05

Chạy Spark Streaming

Khởi động job Spark Streaming để xử lý dữ liệu real-time.

06

Mô phỏng Stream

Khởi động stream simulator để đẩy dữ liệu vào Kafka.

Việc sử dụng Airflow mang lại khả năng **tự động hóa**, **theo dõi trạng thái** từng tác vụ, và **khả năng phục hồi** (retry) khi có lỗi, giúp cho pipeline trở nên mạnh mẽ và đáng tin cậy.

Triển khai trên một máy (Single Machine Deployment)

Các bước chính:

`./scripts/setup.sh`: Thiết lập môi trường cần thiết.

`./scripts/start_kafka.sh`: Khởi động dịch vụ Kafka và Zookeeper.

`python3 data_split.py`: Chia tách bộ dữ liệu.

`spark-submit spark_training.py`: Huấn luyện mô hình Spark ML.

Chạy và Giám sát:

- Mở các terminal riêng để chạy:
  - `spark_streaming.py`
  - `stream_simulator.py`
  - `visualization_consumer.py`
- Sử dụng **Spark UI** và **Kafka UI** để theo dõi các job streaming và luồng dữ liệu, đảm bảo mọi thứ hoạt động trơn tru.

Mô hình triển khai này phù hợp cho giai đoạn phát triển và thử nghiệm, cho phép dễ dàng kiểm soát và debug từng thành phần của hệ thống trên một môi trường cục bộ.

Triển khai đa máy (Multi-Machine Deployment)

Máy 1: Spark Server

Chứa Spark Master và các Spark Worker để xử lý dữ liệu.

Máy 2: Airflow Server

Chạy Airflow Scheduler và Webserver, quản lý các DAG.

Máy 3: Kafka Server

Chứa Kafka Brokers, Zookeeper và Kafka UI.

Máy Demo

Chạy Stream Simulator và ứng dụng Visualization.

Để hệ thống hoạt động hiệu quả trong môi trường sản xuất, cần chú ý cấu hình **IP, ports**, và **firewall** giữa các máy. Đồng thời, các biến môi trường như **SPARK_HOME** và **KAFKA_SERVERS** cần được thiết lập chính xác để các thành phần có thể giao tiếp qua mạng. Đây là bước quan trọng để đảm bảo khả năng mở rộng và hiệu suất của pipeline trong thực tế.

Kết quả mô hình & Hiệu năng Streaming

0.85-0.90

AUC Score

Với mô hình **Random Forest Classifier**, khả năng phân loại tốt.

< 5s

Độ trễ (Latency)

Từ dữ liệu đầu vào đến kết quả dự đoán.

Hệ thống được thiết kế để dễ dàng **mở rộng** bằng cách tăng số lượng Spark workers hoặc Kafka brokers khi cần xử lý lượng dữ liệu lớn hơn. Nhờ sử dụng Kafka và tính năng checkpointing của Spark, pipeline còn có **tính tin cậy cao và khả năng chịu lỗi tốt**, đảm bảo hoạt động liên tục ngay cả khi có sự cố.

Đánh giá, Hạn chế và Hướng phát triển

Thành tựu

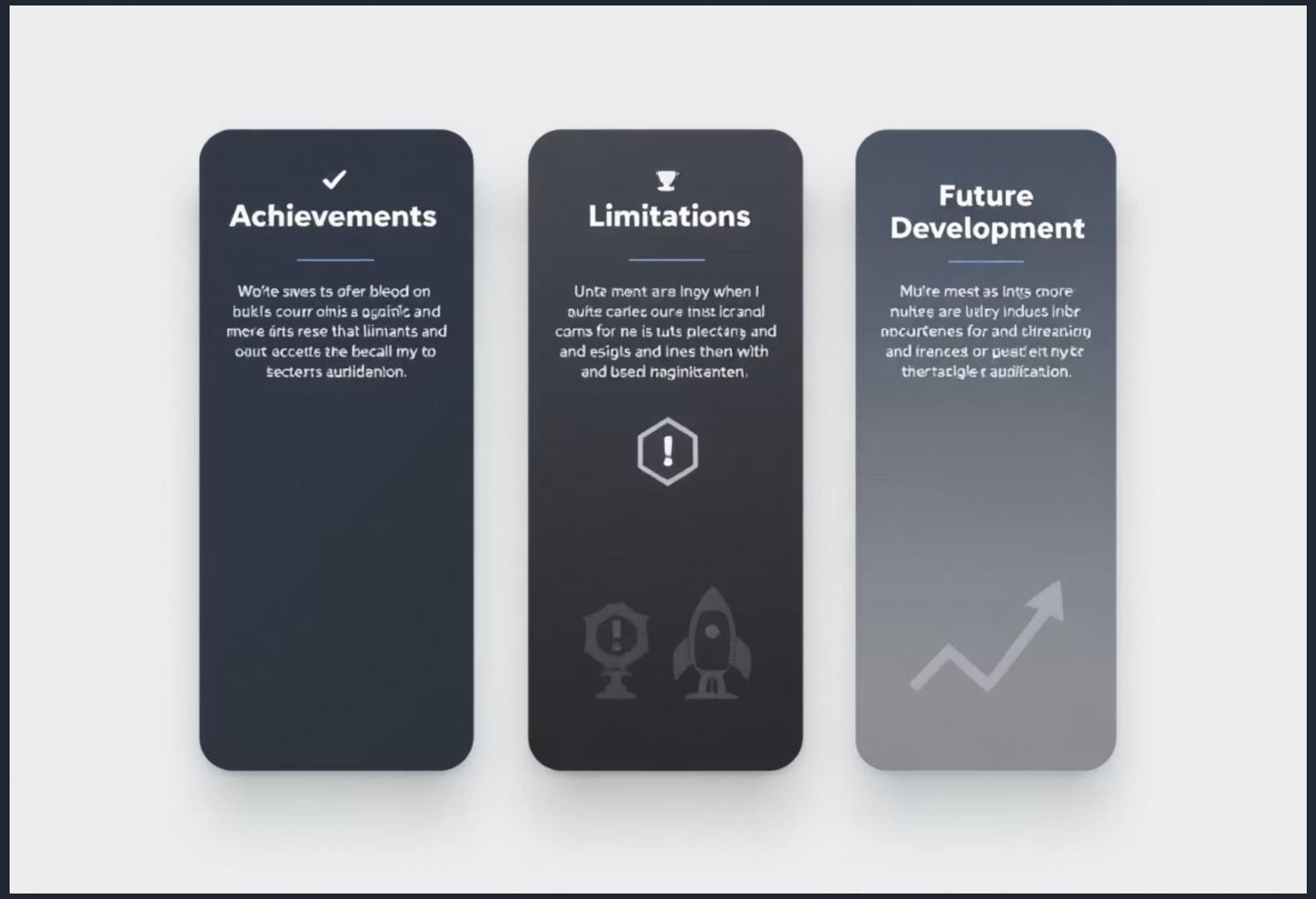
- Pipeline end-to-end, real-time.
- Tích hợp Spark ML, Kafka, Airflow.
- Hỗ trợ triển khai trên một máy & đa máy.

Hạn chế

- Chưa tối ưu mô hình sâu, chưa có A/B testing.
- Chưa có hệ thống monitoring/alerting chuyên nghiệp.

Hướng phát triển

- Thử nghiệm các mô hình khác, thêm feature engineering.
- Tích hợp Prometheus/Grafana, hệ thống cảnh báo.
- Mở rộng Kafka/Spark cho môi trường production.



Về mặt thành tựu, đồ án đã xây dựng được một pipeline hoàn chỉnh, từ tiền xử lý dữ liệu, huấn luyện mô hình, đến streaming và visualization real-time, đồng thời tích hợp được ba công nghệ quan trọng là Spark ML, Kafka và Airflow. Tuy nhiên, hệ thống vẫn còn hạn chế: mô hình chưa được tuning sâu, chưa có A/B testing, và phần monitoring/alerting chưa được triển khai đầy đủ như trong môi trường production. Trong tương lai, có thể mở rộng bằng cách: thử thêm các mô hình khác (XGBoost, deep learning), bổ sung Prometheus/Grafana cho monitoring, và scale out Kafka/Spark để phục vụ dữ liệu lớn thực tế.

Kết luận & Ứng dụng thực tế

Pipeline Hoàn Chỉnh

Đã xây dựng thành công pipeline dự đoán suy tim end-to-end, tích hợp xử lý dữ liệu, dự đoán và trực quan hóa.

Hiệu Năng & Mở Rộng

Hệ thống hoạt động real-time, có khả năng mở rộng linh hoạt và chịu lỗi tốt, sẵn sàng cho môi trường dữ liệu lớn.

- **Giám sát Sức khỏe:** Ứng dụng trong y tế để giám sát bệnh nhân liên tục và cảnh báo sớm các nguy cơ.
- **Xử lý Dữ liệu IoT:** Làm nền tảng cho các hệ thống phân tích dữ liệu từ cảm biến IoT theo thời gian thực.
- **Đánh giá Rủi ro:** Áp dụng cho các bài toán đánh giá rủi ro như phát hiện gian lận trong tài chính hoặc bảo hiểm.

DEMO