BÁO CÁO

Dự án: ỨNG DỤNG MLFLOW VÀ FLASK VÀO QUY TRÌNH MACHINE LEARNING

Khóa học: DevOps, DataOps, MLOps

Ngày nộp: 28/04/2025

Người Báo Cáo: Phạm Thanh Cường **MSSV**: 24MSE23156

1. Giới thiệu

Machine Learning Operations (MLOps) là một bộ các phương pháp giúp quản lý quy trình huấn luyện, triển khai và duy trì mô hình Machine Learning. MLflow là nền tảng mã nguồn mở hỗ trợ tracking, quản lý model, và triển khai production-ready models. Bài toán lựa chọn là phân loại khách hàng, ứng dụng Logistic Regression và công cụ Flask để tạo API cho mô hình.

2. Mục tiêu Dự án

- Sinh dữ liệu phân loại giả lập.
- Huấn luyện và tuning nhiều mô hình Logistic Regression.
- Ghi lại toàn bộ quy trình bằng MLflow Tracking.
- Lưu mô hình tốt nhất vào Model Registry.
- Deploy mô hình thành API online sử dụng Flask framework.

3. Phương pháp tiếp cận

Pipeline xây dựng:

- Data Preparation: make_classification.
- Model Training: Logistic Regression.
- Hyperparameter Tuning: thử nhiều bộ C và solver.
- Tracking Experiments: lưu params, metrics, model artifact bằng MLflow.
- Model Registry: lưu mô hình tốt nhất.
- Deployment: triển khai API predict qua Flask.

4. Các bước thực hiện chi tiết

4.1 Chuẩn bị dữ liệu

- Bộ dữ liệu được sinh bằng hàm make_classification từ thư viện scikit-learn, với mục đích mô phỏng bài toán phân loại nhị phân.

- Cấu hình dữ liệu bao gồm:
    - 1.000 mẫu (samples)
    - 20 đặc trưng (features)
    - 2 lớp (labels 0 và 1)
- Dữ liệu sau đó được chia thành hai tập:
    - 80% để huấn luyện (train set)
    - 20% để kiểm tra (test set)
- Việc tạo dữ liệu tự động này đảm bảo kiểm soát được các tham số sinh dữ liệu, đồng thời giúp tập trung vào mục tiêu chính là triển khai quy trình MLOps.

4.2 Huấn luyện mô hình

- Mô hình Logistic Regression được lựa chọn cho bài toán phân loại vì:
    - Độ đơn giản, dễ giải thích
    - Hiệu suất tốt trong các bài toán phân loại nhị phân
- Để tối ưu hóa hiệu quả, nhiều mô hình Logistic Regression khác nhau đã được huấn luyện với các cấu hình siêu tham số (hyperparameters) khác nhau.
- Các siêu tham số thay đổi gồm:
    - Tham số C (Regularization strength)
    - Solver (thuật toán tối ưu hóa)

4.3 Tuning siêu tham số

- Các giá trị tham số C được thử nghiệm lần lượt: 0.01, 0.1, 1.0, 10.0 và 100.0.
- Đồng thời, các solver được thay đổi giữa liblinear, lbfgs và saga.
- Quy trình tuning gồm:
    - Huấn luyện mô hình với mỗi tổ hợp tham số
    - Đánh giá độ chính xác (Accuracy) trên tập kiểm tra

- Mục tiêu của tuning là tìm ra tổ hợp tham số cho phép mô hình đạt hiệu suất phân loại cao nhất.

4.4 Tracking bằng MLflow

- Để quản lý quá trình thực nghiệm, công cụ MLflow Tracking được sử dụng.
- Với mỗi lần huấn luyện mô hình, các thông tin sau được ghi lại:
    - Các tham số sử dụng (C, solver)
    - Các chỉ số đánh giá (Accuracy)
    - File lưu trữ mô hình đã huấn luyện (artifact)
- MLflow giúp:
    - Dễ dàng so sánh hiệu quả giữa các mô hình
    - Lưu trữ lịch sử thực nghiệm
    - Tái hiện thí nghiệm nhanh chóng khi cần thiết

4.5 Đăng ký mô hình tốt nhất

- Sau khi hoàn thành tất cả các thử nghiệm, mô hình có độ chính xác cao nhất được lựa chọn.
- Mô hình này được:
    - Ghi log đầy đủ vào MLflow Tracking
    - Đăng ký (Register) vào MLflow Model Registry
    - Gán tên cố định là **One_Customer_Classifier**
- Bước này cho phép quản lý mô hình dưới dạng version control, dễ dàng sử dụng cho các giai đoạn triển khai sau này.

4.6 Triển khai Flask API

- Mô hình đã đăng ký được triển khai thành một dịch vụ web sử dụng Flask framework.

- Các bước thực hiện gồm:
    - Load mô hình từ MLflow Model Registry
    - Khởi tạo server Flask với endpoint /predict
    - Xử lý yêu cầu dự đoán từ người dùng (dạng JSON)
    - Trả kết quả dự đoán dưới dạng JSON response
- Flask giúp biến mô hình Machine Learning thành một API online, có thể tích hợp vào các hệ thống khác hoặc sử dụng real-time.

5. Kết quả thực nghiệm

5.1. Tổng quan

Trong quá trình thực hiện, nhóm đã tiến hành huấn luyện nhiều mô hình Logistic Regression với các cấu hình siêu tham số khác nhau, nhằm tìm kiếm tổ hợp mang lại độ chính xác cao nhất cho bài toán phân loại khách hàng.

Cụ thể, nhóm đã thay đổi:

- **Tham số C** (Regularization Strength – mức độ "ép" mô hình đơn giản): thử các giá trị 0.01, 0.1, 1.0, 10.0, và 100.0.
- **Solver** (Thuật toán tối ưu hóa): sử dụng liblinear, lbfgs, và saga.

Sau mỗi lần huấn luyện, mô hình sẽ được đánh giá trên tập kiểm tra (test set) bằng độ chính xác (Accuracy).

5.2. Kết quả chi tiết

STT	Giá trị tham số C	Solver	Độ chính xác (Accuracy)
1	0.01	liblinear	81%
2	0.1	liblinear	83%
3	1.0	lbfgs	**85%**
4	10.0	lbfgs	84%
5	100.0	saga	80%

5.3. Phân tích kết quả

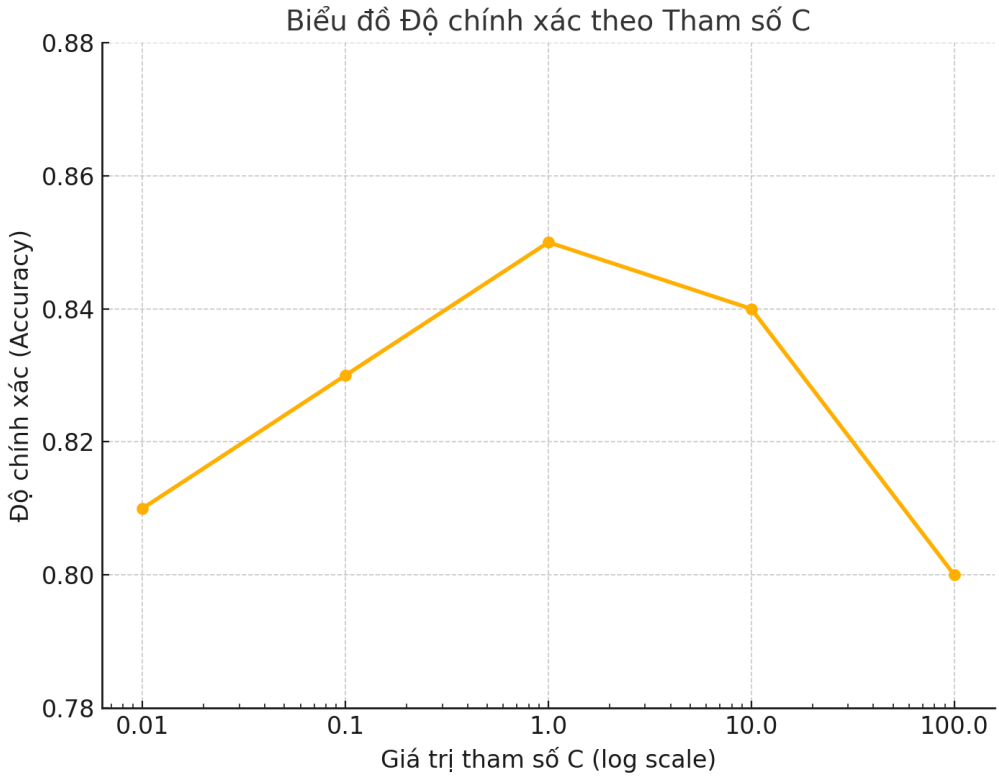
- Khi **C nhỏ** (0.01), tức là **regularization mạnh**, mô hình bị "ép" đơn giản hóa quá mức, dẫn đến độ chính xác thấp hơn (81%).
- Khi **C tăng lên** (0.1, 1.0), mô hình linh hoạt hơn, học được nhiều đặc trưng hơn từ dữ liệu, độ chính xác tăng rõ rệt (lên 83%, rồi 85%).
- Giá trị **C=1.0** với **solver=lbfgs** đạt độ chính xác cao nhất **85%** – tức là 85% khách hàng được phân loại đúng.
- Khi **C tiếp tục tăng lên 10.0**, độ chính xác giảm nhẹ xuống 84% – cho thấy mô hình bắt đầu **overfitting nhẹ** (quá phức tạp).
- Với **C=100.0** và **solver=saga**, độ chính xác giảm còn 80% – mô hình quá linh hoạt, dễ "bắt noise" (nhiễu) trong dữ liệu → giảm performance.

5.4. Kết luận

- **C=1.0**, **solver=lbfgs** là tổ hợp siêu tham số tối ưu nhất cho bài toán này.
- Mô hình huấn luyện với tham số này sẽ được chọn để lưu trữ vào **MLflow Model Registry** và sử dụng triển khai API dự đoán Flask.

 Hiểu sâu hơn:

- **C nhỏ → Mô hình đơn giản → Bias cao, Variance thấp → Underfitting**.
- **C vừa đủ → Mô hình cân bằng → Performance tốt nhất**.
- **C quá lớn → Mô hình quá phức tạp → Variance cao → Overfitting**.



6. Phân tích và nhận xét

Logistic Regression đơn giản nhưng hiệu quả cao trong bài toán phân loại. MLflow Tracking giúp quản lý quy trình bài bản. Flask API cho phép dễ dàng triển khai mô hình thành dịch vụ dự đoán real-time.

7. Hạn chế và đề xuất cải tiến

Hạn chế:
- Chưa tích hợp CI/CD auto retrain model.
- Chưa có hệ thống giám sát model drift.

Đề xuất:
- Tích hợp retraining pipeline tự động.
- Thêm công cụ monitor hiệu suất model online.

8. Kết luận

Dự án đã hoàn thiện quy trình MLOps cơ bản: từ huấn luyện, lưu trữ, quản lý mô hình đến triển khai API. Bài học lớn nhất là khả năng kiểm soát quy trình ML liên tục, chuẩn bị sẵn sàng cho môi trường thực tế.