ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

CÔNG NGHỆ DEVOPS VÀ ỨNG DỤNG
NT548.P21

Báo cáo cuối kỳ

MLOPS: TỰ ĐỘNG HÓA TRIỂN KHAI PHÂN TÍCH CẢM XÚC VỚI SAGEMAKER, GITHUB ACTIONS VÀ CLOUDFORMATION

Sinh viên :
Lê Bình Nguyên - 22520969
Đặng Hữu Phát - 22521065
Châu Thế Vĩ - 22521653

Giảng viên hướng dẫn :
ThS. Lê Anh Tuấn

Ngày 27 tháng 5 năm 2025

LỜI CẢM ƠN

Nhóm chúng em xin bày tỏ lòng biết ơn sâu sắc đến thầy **Lê Anh Tuấn**, giảng viên Khoa Mạng máy tính và Truyền thông, Trường Đại học Công nghệ Thông tin, Đại học Quốc gia Thành phố Hồ Chí Minh. Thầy đã tận tình giảng dạy và hướng dẫn nhóm trong học phần *Công nghệ DevOps và Ứng dụng*, giúp chúng em xây dựng nền tảng kiến thức vững chắc để hoàn thành đồ án này.

Với sự hướng dẫn chi tiết, tận tình của thầy, chúng em không chỉ nắm vững các nguyên tắc cơ bản mà còn học được cách ứng dụng kiến thức vào thực tiễn. Đây là động lực to lớn giúp chúng em tự tin hoàn thành đồ án một cách tốt nhất.

Bên cạnh đó, nhóm cũng xin gửi lời cảm ơn chân thành đến các thành viên trong nhóm, gồm:

- **Lê Bình Nguyên** – 22520969
- **Đặng Hữu Phát** – 22521065
- **Châu Thế Vĩ** – 22521653

Các bạn đã cùng nhau đồng hành, không ngừng nỗ lực và cống hiến để đạt được những thành quả tốt đẹp như ngày hôm nay. Tinh thần hợp tác và hỗ trợ lẫn nhau trong suốt thời gian thực hiện đồ án là nguồn động lực lớn lao để cả nhóm cùng vượt qua những khó khăn và thử thách.

Dẫu đã cố gắng hết sức, nhưng trong quá trình thực hiện đồ án, khó có thể tránh khỏi những thiếu sót. Nhóm rất mong nhận được những ý kiến đóng góp quý báu từ thầy để có thể hoàn thiện hơn trong tương lai. Mọi ý kiến đóng góp hoặc thắc mắc, xin vui lòng liên hệ qua email của các thành viên nhóm:

- 22520969gm.uit.edu.vn (Lê Bình Nguyên)
- 22521065@gm.uit.edu.vn (Đặng Hữu Phát)
- 22521653@gm.uit.edu.vn (Châu Thế Vĩ)

Thông tin chi tiết về đồ án và mã nguồn được lưu tại GitHub Repostitory:

GitHub Repository.

Một lần nữa, nhóm chúng em xin chân thành cảm ơn!

Nhóm thực hiện: **Nhóm 21**.

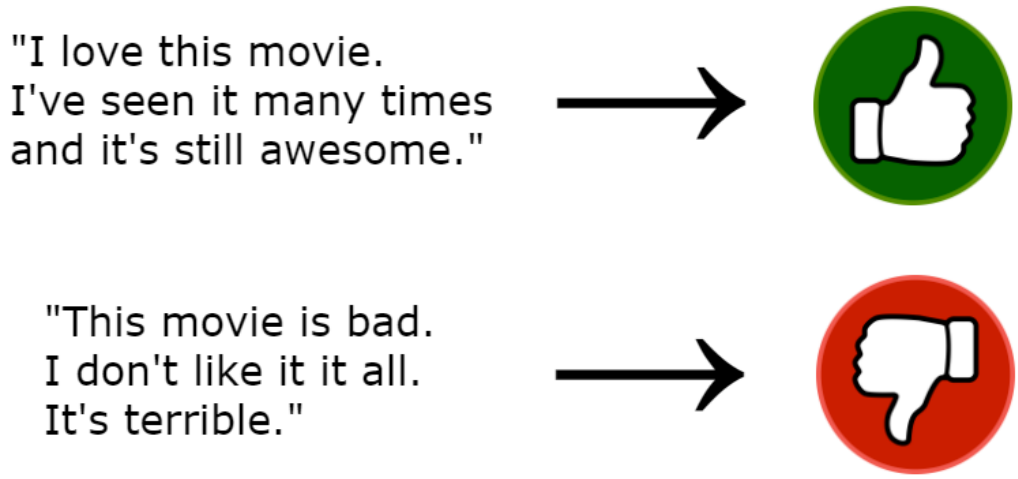
Mục lục

Chương 1

Giới thiệu

1.1 Bối cảnh và Động lực

Trong thời đại số hóa hiện nay, sự bùng nổ của các nền tảng trực tuyến như mạng xã hội, diễn đàn và các trang web đánh giá đã tạo ra một khối lượng dữ liệu văn bản khổng lồ. Đặc biệt, các bình luận đánh giá phim là nguồn thông tin quan trọng, phản ánh ý kiến, cảm xúc và thái độ của khán giả đối với các tác phẩm điện ảnh. Tuy nhiên, việc phân tích thủ công hàng triệu bình luận này không chỉ tốn kém thời gian mà còn dễ xảy ra sai sót, khiến cho các phương pháp tự động hóa trở thành một nhu cầu cấp thiết.



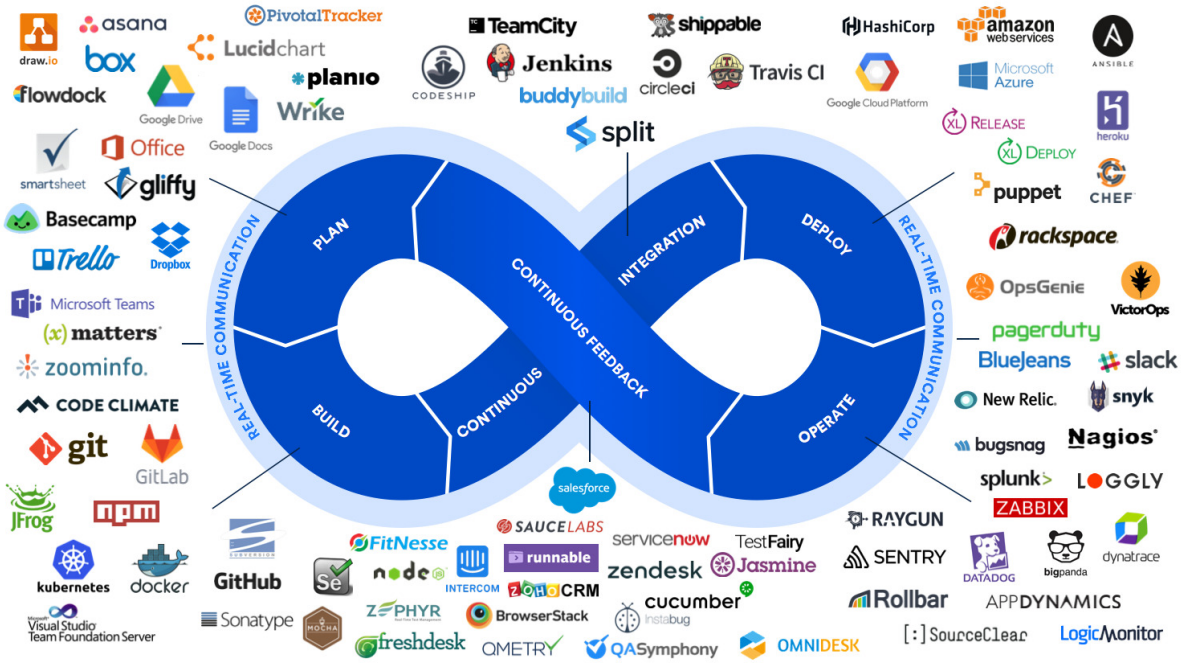
Hình 1.1: Hình ảnh minh họa thể hiện hai quan điểm trái ngược về một bộ phim, tương ứng với đánh giá tích cực và tiêu cực.

Phân tích cảm xúc (sentiment analysis) là một kỹ thuật thuộc lĩnh vực xử lý ngôn ngữ tự nhiên (NLP), cho phép xác định và phân loại cảm xúc trong văn bản, chẳng hạn như tích cực, tiêu cực hoặc trung lập. Trong bối cảnh đánh giá phim, phân tích cảm xúc không chỉ hỗ trợ các nhà sản xuất và phân phối nắm bắt phản hồi của khán giả mà còn giúp người xem đưa ra quyết định dựa trên ý kiến cộng đồng. Tuy nhiên, việc triển khai một hệ thống phân tích cảm xúc hiệu quả đòi hỏi phải vượt qua nhiều thách thức, bao gồm xử lý dữ liệu lớn, đảm bảo tính nhất quán giữa các giai đoạn phát triển và vận hành, cũng như duy trì hiệu suất trong môi trường thực tế.

Để giải quyết những vấn đề này, đồ án áp dụng các phương pháp luận hiện đại như **DevOps** và **MLOps**, kết hợp với các công nghệ tiên tiến như **Amazon SageMaker**, **GitHub Actions** và **AWS CloudFormation**. Mục tiêu là xây dựng một hệ thống tự động hóa toàn diện cho phân tích cảm xúc của các bình luận đánh giá phim, vừa giảm thiểu sai sót thủ công vừa tăng tốc độ triển khai.

1.2 Giới thiệu về DevOps và MLOps

1.2.1 DevOps



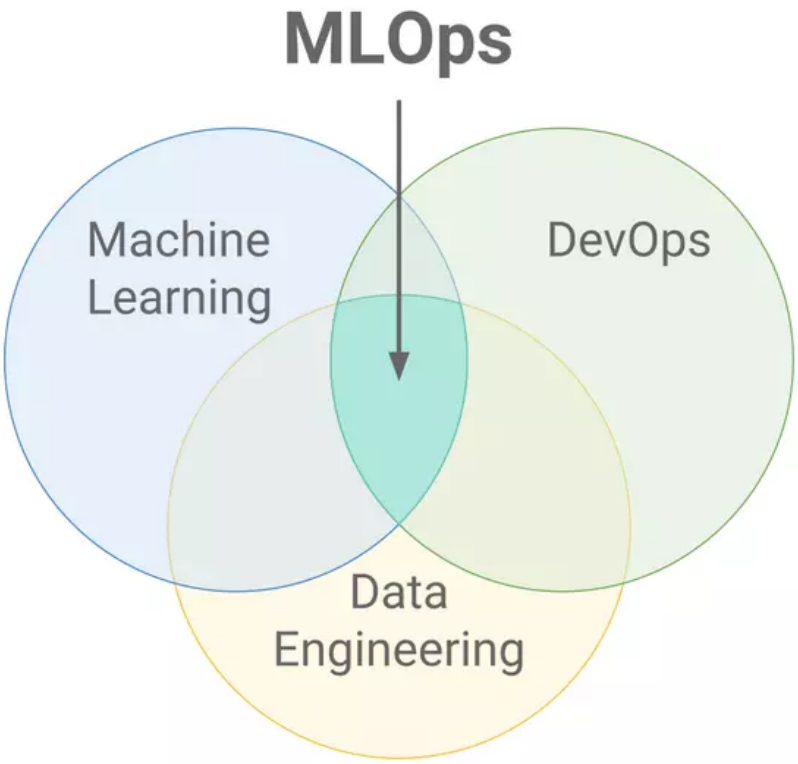
Hình 1.2: Hình ảnh này minh họa vòng đời DevOps dưới dạng một vòng lặp vô cực, với các giai đoạn chính bao gồm Kế hoạch (Plan), Xây dựng (Build), Tích hợp liên tục/Phản hồi liên tục (Continuous Integration/Continuous Feedback), và Vận hành (Operate).

DevOps là một phương pháp luận kết hợp giữa phát triển phần mềm (Development) và vận hành hệ thống (Operations), nhằm tối ưu hóa chu kỳ phát triển và triển khai. Các nguyên tắc chính của DevOps bao gồm:

- **Tích hợp liên tục (CI)**: Tự động hóa việc kiểm tra và tích hợp mã nguồn từ nhiều nhà phát triển.
- **Triển khai liên tục (CD)**: Đảm bảo ứng dụng được triển khai nhanh chóng và đáng tin cậy lên các môi trường khác nhau.
- **Tự động hóa**: Giảm thiểu các thao tác thủ công để tăng tính nhất quán và hiệu quả.
- **Giám sát**: Theo dõi hiệu suất hệ thống và cải thiện liên tục dựa trên dữ liệu thực tế.

DevOps giúp các nhóm phát triển đẩy nhanh tiến độ, nâng cao chất lượng phần mềm và tối ưu hóa quy trình vận hành.

1.2.2 MLOps



Hình 1.3: Sơ đồ minh họa MLOps – sự kết hợp giữa Machine Learning, DevOps và Data Engineering trong quy trình phát triển AI.

MLOps là một nhánh mở rộng của DevOps, được thiết kế đặc biệt để quản lý vòng đời của các mô hình học máy. MLOps giải quyết các thách thức như:

- **Quản lý dữ liệu**: Xử lý và lưu trữ dữ liệu huấn luyện một cách hiệu quả.
- **Phân phiên bản**: Theo dõi và quản lý các phiên bản khác nhau của mô hình.
- **Huấn luyện và triển khai tự động**: Tự động hóa các bước huấn luyện, đánh giá và triển khai mô hình.
- **Giám sát hiệu suất**: Đảm bảo mô hình duy trì hiệu suất ổn định sau khi triển khai.

MLOps mang lại sự kết hợp giữa các thực hành DevOps và yêu cầu đặc thù của học máy, giúp triển khai các mô hình một cách đáng tin cậy và dễ bảo trì.

1.3 Mục tiêu của đồ án

Đồ án này tập trung vào việc xây dựng một hệ thống tự động hóa cho bài toán phân tích cảm xúc của các bình luận đánh giá phim, với các mục tiêu cụ thể:

- Thiết lập một **pipeline MLOps** tự động hóa các giai đoạn huấn luyện, đóng gói và triển khai mô hình.
- Phát triển một **API** để dự đoán cảm xúc theo thời gian thực từ các bình luận phim.
- Xây dựng một **ứng dụng web** tích hợp API, cung cấp giao diện thân thiện cho người dùng.
- Đảm bảo hệ thống đáp ứng các tiêu chí về **bảo mật**, **khả năng mở rộng** và **dễ bảo trì**.

Hệ thống không chỉ giải quyết bài toán phân tích cảm xúc mà còn được thiết kế để có thể tái sử dụng và mở rộng cho các ứng dụng học máy khác.

1.4 Ý nghĩa và Giá trị của đồ án

Đồ án mang lại nhiều giá trị thực tiễn và ý nghĩa quan trọng:

- Ứng dụng thực tiễn phân tích cảm xúc trong việc đánh giá phim, hỗ trợ cả khán giả và nhà sản xuất.
- Kết hợp lý thuyết MLOps với thực hành để xây dựng một hệ thống học máy hoàn chỉnh.
- Đặt nền móng cho các đồ án học máy phức tạp hơn, với khả năng mở rộng và tái sử dụng cao.

Phần giới thiệu này làm rõ bối cảnh, mục tiêu và tầm quan trọng của đồ án, đồng thời nhấn mạnh vai trò của công nghệ hiện đại trong việc giải quyết các vấn đề thực tế.

Chương 2

Các công cụ và nền tảng sử dụng

2.1 Hạ tầng đám mây (Cloud Infrastructure)

2.1.1 Amazon SageMaker

Amazon SageMaker là một nền tảng toàn diện do Amazon Web Services (AWS) cung cấp, hỗ trợ xây dựng, huấn luyện, tinh chỉnh và triển khai mô hình học máy một cách hiệu quả ở quy mô lớn. Trong đồ án phân tích cảm xúc từ dữ liệu IMDB, SageMaker đóng vai trò quan trọng trong việc huấn luyện mô hình LSTM và quản lý các phiên bản mô hình, đảm bảo hiệu suất cao và khả năng mở rộng. Nền tảng này cung cấp các tính năng như môi trường huấn luyện linh hoạt, tích hợp chặt chẽ với hệ sinh thái AWS và khả năng triển khai mô hình thành các *endpoint* cho suy luận thời gian thực.

Hình 2.1: Amazon SageMaker

SageMaker được sử dụng trong đồ án này để thực hiện hai nhiệm vụ chính. Trước tiên, nó hỗ trợ quá trình huấn luyện mô hình LSTM thông qua các bước cụ thể:

- Tải tập dữ liệu IMDB từ Amazon S3, nơi dữ liệu được lưu trữ an toàn và dễ truy cập.
- Huấn luyện mô hình trên các instance GPU mạnh mẽ do SageMaker cung cấp, giúp tăng tốc độ xử lý và tối ưu hóa hiệu suất.
- Lưu trữ artifact của mô hình (bao gồm trọng số và cấu hình) sau khi huấn luyện để tái sử dụng hoặc triển khai.

Thứ hai, SageMaker hỗ trợ quản lý phiên bản mô hình một cách hiệu quả:

- Lưu trữ và quản lý các phiên bản mô hình sau khi huấn luyện.
- Triển khai mô hình thành endpoint, cho phép ứng dụng bên ngoài gửi yêu cầu dự đoán và nhận kết quả ngay lập tức.

Việc sử dụng SageMaker mang lại nhiều lợi ích nổi bật cho đồ án. Nó cho phép mở rộng dễ dàng nhờ khả năng tự động điều chỉnh tài nguyên theo nhu cầu, giúp xử lý khối lượng dữ liệu lớn hơn trong tương lai mà không cần thay đổi quy trình. SageMaker cũng tích hợp mượt mà với các dịch vụ AWS như S3, ECR và Lambda, tạo ra một quy trình làm việc liền mạch và hiệu quả. Ngoài ra, nền tảng này giảm thiểu gánh nặng quản lý hạ tầng, cho phép nhóm phát triển tập trung vào việc cải thiện mô hình thay vì lo lắng về cấu hình máy chủ hay tối ưu hóa tài nguyên. Nhờ những ưu điểm này, SageMaker không chỉ nâng cao hiệu suất của đồ án mà còn đảm bảo tính linh hoạt và khả năng mở rộng trong dài hạn.

2.1.2 Amazon S3

Hình 2.2: Amazon S3

Amazon Simple Storage Service (S3) là dịch vụ lưu trữ đối tượng của AWS, nổi bật với khả năng mở rộng gần như vô hạn, độ tin cậy cao và khả năng tích hợp mạnh mẽ với các dịch vụ khác.

S3 được sử dụng trong đồ án để:

- Lưu trữ tập dữ liệu huấn luyện.
- Lưu kết quả huấn luyện và mô hình đã được huấn luyện.
- Chia sẻ tài nguyên giữa các bước trong pipeline.

2.1.3 Amazon ECR

Amazon Elastic Container Registry (ECR) là một dịch vụ được AWS cung cấp để quản lý, lưu trữ và chia sẻ các Docker image một cách an toàn, hiệu quả và dễ tích hợp vào các quy trình DevOps. Dịch vụ này hỗ trợ tích hợp trực tiếp với các công cụ như AWS IAM, Amazon ECS, Amazon SageMaker và CI/CD pipelines, giúp tăng tốc độ phát triển và triển khai ứng dụng.

Hình 2.3: Amazon ECR

Trong đồ án này, Amazon ECR đóng vai trò quan trọng trong việc container hoá và chuẩn hóa môi trường thực thi. Cụ thể, ECR được sử dụng để:

- Lưu trữ image của container custom chứa mã nguồn huấn luyện và inference.
- Tích hợp với SageMaker để đảm bảo môi trường huấn luyện được kiểm soát và tái sử dụng.

Việc sử dụng Amazon ECR không chỉ giúp tiêu chuẩn hóa môi trường triển khai mô hình học máy mà còn đóng vai trò trung tâm trong việc kết nối các dịch vụ như SageMaker, Lambda và CloudFormation, tạo nên một hệ sinh thái liền mạch và linh hoạt cho MLOps.

2.1.4 AWS Lambda

AWS Lambda là một dịch vụ tính toán không máy chủ (serverless) do Amazon Web Services (AWS) cung cấp. Nó cho phép người dùng chạy mã nguồn dưới dạng các hàm (functions) mà không cần phải quản lý hoặc cấu hình bất kỳ máy chủ vật lý hay ảo nào. Với Lambda, người dùng có thể tập trung hoàn toàn vào việc phát triển ứng dụng mà không cần lo lắng về quản lý cơ sở hạ tầng như cấp phát tài nguyên, mở rộng hệ thống, hay bảo trì máy chủ.

Hình 2.4: Amazon Lambda

Trong đồ án, Lambda đảm nhận vai trò:

- Xử lý các yêu cầu API.
- Định dạng đầu vào trước khi gửi đến SageMaker Endpoint.
- Nhận kết quả suy luận và trả về client.

Ví dụ:

- Người dùng gửi đánh giá phim qua API Gateway: "Phim hay!".
- Lambda xử lý, gửi SageMaker dự đoán "tích cực", rồi trả kết quả về client

Lambda là cầu nối quan trọng trong đồ án, kết nối API Gateway và SageMaker, mang lại hiệu quả, khả năng mở rộng và tiết kiệm chi phí cho hệ thống.

2.1.5 Amazon API Gateway

Hình 2.5: Amazon API Gateway

Amazon API Gateway là dịch vụ giúp tạo, quản lý, bảo mật và giám sát các API RESTful hoặc WebSocket một cách dễ dàng và hiệu quả. Dịch vụ này cho phép tích hợp liền mạch giữa ứng dụng frontend và các dịch vụ backend trên AWS, đồng thời hỗ trợ các cơ chế kiểm soát truy cập và giám sát chuyên sâu.

Trong đồ án, API Gateway đóng vai trò là điểm truy cập chính cho người dùng cuối, với các chức năng cụ thể như:

- Làm cầu nối giữa frontend và Lambda: Nhận yêu cầu từ người dùng qua giao diện web và chuyển tiếp đến AWS Lambda để xử lý và suy luận.
- Bảo mật truy cập: Hạn chế quyền truy cập bằng cách cấu hình các phương thức xác thực phù hợp.
- Theo dõi và ghi log: Tích hợp với CloudWatch để giám sát hiệu suất API và phân tích các request từ client.

Việc sử dụng API Gateway giúp chuẩn hóa giao tiếp trong hệ thống, đảm bảo khả năng mở rộng và giám sát hiệu quả trong toàn bộ pipeline triển khai mô hình học máy.

2.2 Khung học máy và Thư viện

2.2.1 Python

Hình 2.6: Python

Ngôn ngữ lập trình chính được sử dụng xuyên suốt toàn bộ đồ án, nhờ vào:

- Cú pháp đơn giản, dễ đọc, dễ duy trì.
- Thư viện học máy và xử lý dữ liệu phong phú.
- Hỗ trợ tốt cho container hóa và tích hợp với các dịch vụ AWS.

2.2.2 PyTorch

Hình 2.7: Pytorch

PyTorch là một framework học sâu mã nguồn mở, nổi bật với:

- Cơ chế mô hình hóa động giúp dễ dàng debug.
- Tối ưu tốt trên GPU.
- Cộng đồng mạnh và tài liệu đầy đủ.

Trong đồ án, PyTorch được sử dụng để xây dựng mô hình LSTM phân tích cảm xúc.

2.2.3 scikit-learn

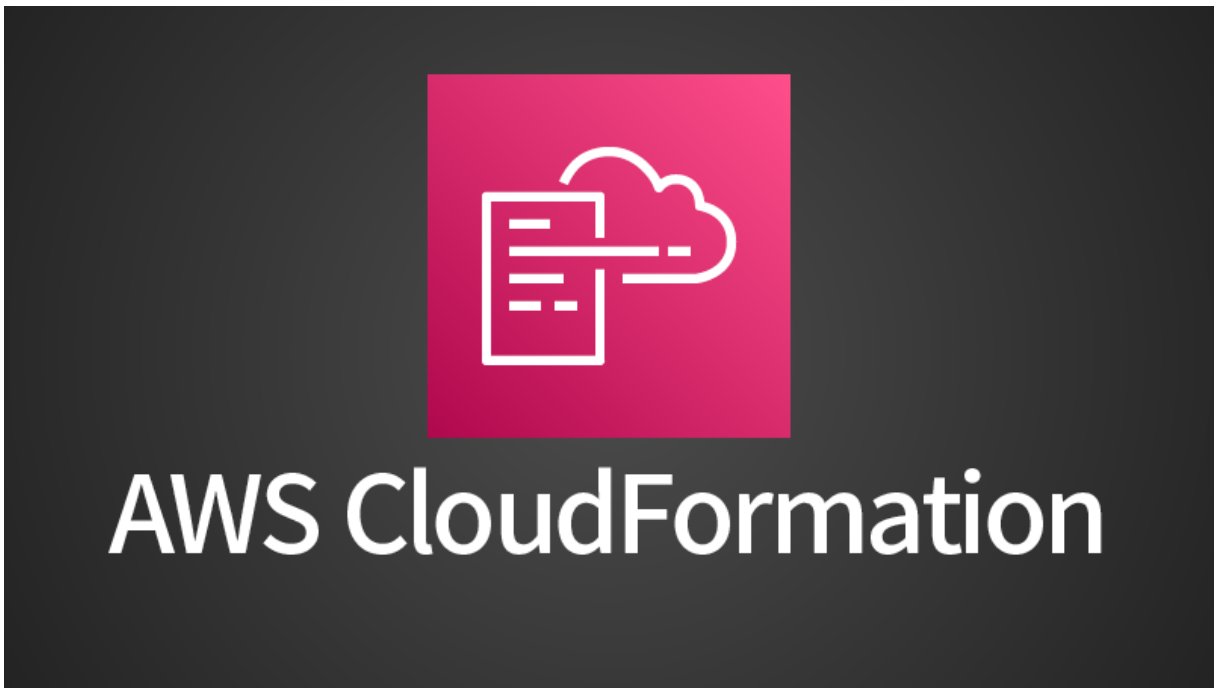
Hình 2.8: Scikit-learn

Thư viện cổ điển cho các thuật toán học máy, tiền xử lý và đánh giá mô hình.

Ứng dụng trong đồ án:

- Chuẩn hóa dữ liệu đầu vào.
- Đánh giá mô hình với các chỉ số như Accuracy, F1-score.
- Hỗ trợ kiểm thử và so sánh nhiều pipeline.

2.3 Quản lý hạ tầng bằng mã (IaC)



Hình 2.9: Amazon CloudFormation

Trong đồ án, nhóm sử dụng AWS CloudFormation để quản lý hạ tầng theo hướng "Infrastructure as Code"(IaC). Đây là dịch vụ cho phép định nghĩa toàn bộ tài nguyên

AWS như API Gateway, Lambda, S3, SageMaker Endpoint,... thông qua các tệp YAML hoặc JSON. Việc này giúp triển khai hạ tầng một cách nhanh chóng, nhất quán và có thể theo dõi bằng phiên bản mã nguồn.

Việc áp dụng CloudFormation trong dự án mang lại nhiều lợi ích thiết thực:

- Giảm lỗi cấu hình thủ công: Hạn chế rủi ro do thao tác tay nhờ định nghĩa tài nguyên bằng mã.
- Dễ dàng mở rộng và tái sử dụng: Có thể sử dụng lại cùng một template để triển khai nhiều môi trường khác nhau như dev, test, prod.
- Tự động hóa toàn bộ môi trường: Khi kết hợp với CI/CD, toàn bộ hạ tầng được thiết lập tự động chỉ với một file cấu hình duy nhất.

Tóm lại, CloudFormation giúp đảm bảo hệ thống được triển khai đồng nhất, dễ bảo trì và mở rộng, là nền tảng quan trọng trong thực hiện MLOps chuyên nghiệp.

2.4 Tự động hóa CI/CD

Hình 2.10: GitHub Actions

Nhóm triển khai tự động hóa CI/CD bằng GitHub Actions. GitHub Actions là nền tảng tự động hóa quy trình ngay trong kho mã nguồn GitHub. Trong đồ án, Actions giúp:

- Kiểm thử mã tự động khi có pull request.
- Build và đẩy Docker image lên ECR.
- Triển khai mô hình và API.
- Deploy giao diện web lên S3.

2.5 Container hoá

Hình 2.11: GitHub Actions

Docker là công cụ container hóa ứng dụng phổ biến, được sử dụng để:

- Đóng gói môi trường huấn luyện và inference.
- Tái sử dụng mô hình dễ dàng qua các bước.
- Đảm bảo môi trường thống nhất giữa dev và prod.

2.6 Giám sát và Ghi log

Hình 2.12: Amazon CloudWatch

Amazon CloudWatch là dịch vụ giám sát và quan sát hệ thống toàn diện được cung cấp bởi AWS. CloudWatch cho phép thu thập và phân tích log, theo dõi hiệu suất tài nguyên, cũng như cấu hình cảnh báo theo thời gian thực, giúp phát hiện và xử lý sự cố nhanh chóng.

Trong khuôn khổ đồ án, CloudWatch được sử dụng để:

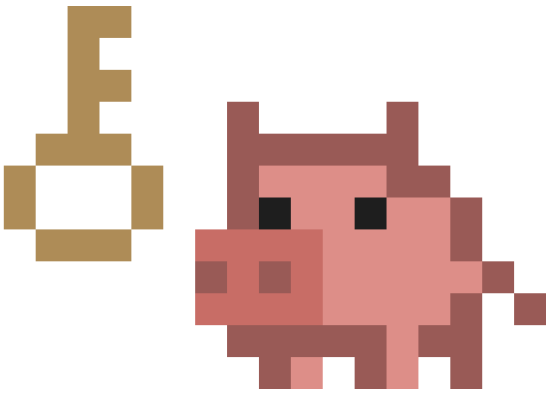
- Theo dõi quá trình huấn luyện trên SageMaker: Giám sát log và thông số từ các training job, giúp phát hiện sớm các vấn đề về hiệu suất hoặc lỗi xử lý.
- Ghi log từ Lambda và các lỗi hệ thống: Toàn bộ các request/response và lỗi từ hàm Lambda đều được ghi lại trong CloudWatch, hỗ trợ quá trình debug và kiểm tra hoạt động hệ thống.
- Cảnh báo và phân tích sự cố: Cho phép thiết lập cảnh báo dựa trên điều kiện cụ thể (ví dụ: lỗi vượt ngưỡng, thời gian phản hồi cao), từ đó chủ động xử lý trước khi hệ thống bị gián đoạn.

Nhờ tích hợp chặt chẽ với các dịch vụ AWS khác, CloudWatch đóng vai trò quan trọng trong việc đảm bảo hệ thống hoạt động ổn định, minh bạch và dễ theo dõi trong suốt vòng đời triển khai mô hình.

2.7 Phân tích bảo mật mã nguồn

Trong quá trình triển khai ứng dụng học máy, bảo mật mã nguồn là yếu tố quan trọng nhằm đảm bảo an toàn cho hệ thống và dữ liệu người dùng. Đồ án sử dụng hai công cụ phổ biến là TruffleHog và Bandit để kiểm tra các lỗ hổng bảo mật trong mã Python và phát hiện thông tin nhạy cảm bị vô tình đẩy lên Git.

2.7.1 TruffleHog



Hình 2.13: TruffleHog

TruffleHog là công cụ quét toàn bộ lịch sử Git nhằm phát hiện các chuỗi nhạy cảm như API key, secret token, mật khẩu hoặc thông tin định danh bị lộ. Công cụ sử dụng thuật toán regex và entropy để xác định các chuỗi có độ ngẫu nhiên cao – đặc trưng của khóa bảo mật.

Trong đồ án, TruffleHog được tích hợp vào pipeline CI để kiểm tra trước khi merge code, giúp phát hiện sớm và ngăn ngừa nguy cơ rò rỉ dữ liệu bảo mật.

2.7.2 Bandit

Hình 2.14: Bandit

Bandit là công cụ phân tích tĩnh mã nguồn Python nhằm phát hiện các lỗ hổng bảo mật phổ biến. Công cụ này kiểm tra mã dựa trên một tập hợp quy tắc, cung cấp cảnh báo kèm theo mức độ nghiêm trọng và đề xuất cách khắc phục.

Một số lỗi tiêu biểu được Bandit phát hiện bao gồm:

- Sử dụng hàm `eval()` hoặc `exec()` không an toàn, dễ dẫn đến thực thi mã độc.
- Xử lý file hoặc đầu vào người dùng không kiểm tra đúng cách, gây nguy cơ injection hoặc mất dữ liệu.
- Cấu hình sai các thư viện hoặc thiếu kiểm soát exception, gây lộ thông tin hoặc khiến hệ thống không ổn định.

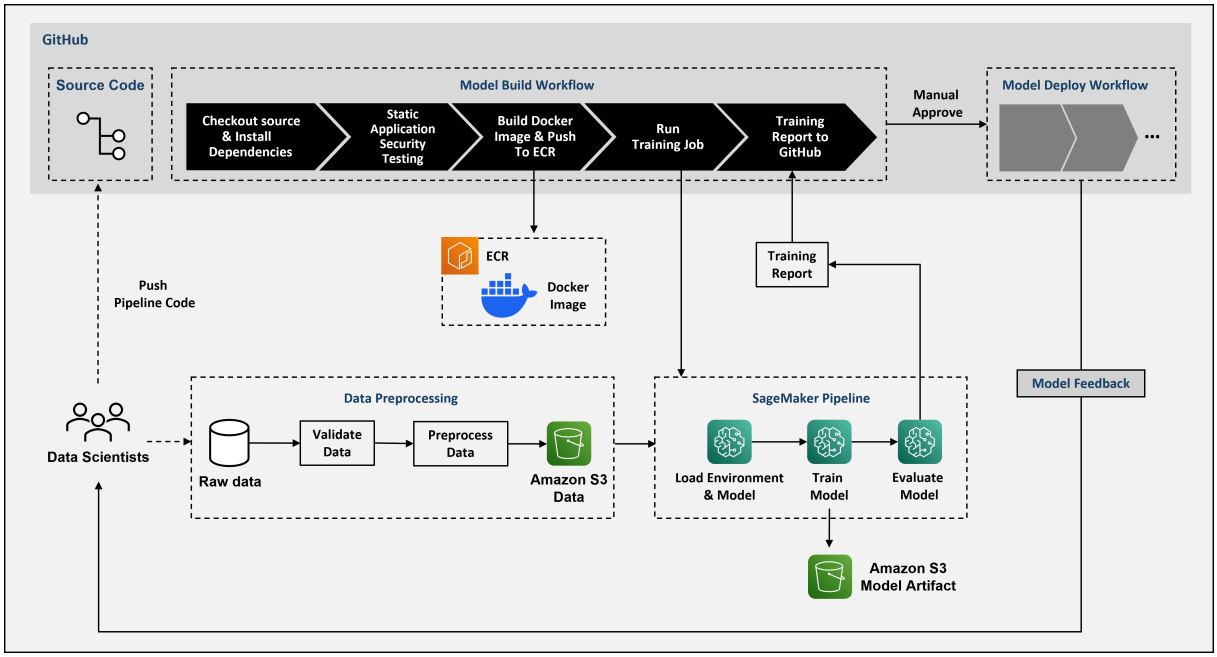
Việc kết hợp TruffleHog và Bandit giúp đồ án nâng cao mức độ an toàn trong toàn bộ vòng đời phát triển, đảm bảo mã nguồn không chỉ hoạt động tốt mà còn tuân thủ các tiêu chuẩn bảo mật cần thiết.

Chương 3

Kiến trúc hệ thống

Trong chương này, nhóm sẽ trình bày cấu trúc phức tạp của hệ thống MLOps và cách Amazon SageMaker hỗ trợ huấn luyện và triển khai mô hình học máy, GitHub Actions tự động hóa quy trình làm việc, cùng với AWS CloudFormation quản lý cơ sở hạ tầng và một số công cụ khác cần thiết cho quá trình thực hiện.

3.1 Quá trình huấn luyện, đánh giá mô hình



Hình 3.1: Pipeline huấn luyện và đánh giá mô hình

Hệ thống CI/CD bao gồm các thành phần chính như sau:

- **Nguồn mã (GitHub)**: Nơi chứa mã nguồn huấn luyện mô hình và mã workflow GitHub Actions.
- **Docker + Amazon ECR**: Dùng để đóng gói môi trường huấn luyện và đẩy lên Amazon ECR.

- **Amazon SageMaker**: Dịch vụ chính để train và evaluate mô hình.
- **Amazon S3**: Lưu trữ dữ liệu huấn luyện và artifacts mô hình sau huấn luyện.
- **Bandit, TruffleHog**: Các công cụ kiểm tra bảo mật và chất lượng mã nguồn.
- **CML (Continuous Machine Learning)**: Để gửi báo cáo huấn luyện về pull request.

3.1.1 Các bước chi tiết trong pipeline

Pipeline được chia thành các giai đoạn chính như sau:

1. **Push Mã Nguồn**
   Khi nhà khoa học dữ liệu tạo pull request lên nhánh `dev`, workflow trong GitHub sẽ được kích hoạt:

```
on:
  pull_request:
    branches: [dev]
```

2. **Thiết lập môi trường CI**
   2.1. Checkout mã nguồn từ GitHub repository.
   2.2. Thiết lập môi trường Python 3.9 và cài đặt các dependencies cần thiết:
   - `sagemaker`, `boto3`, `scikit-learn`, `nltk`, `pandas`, `trufflehog`, `bandit`, v.v.

3. **Phân tích bảo mật và chất lượng mã nguồn**
   - **Bandit**: Phân tích bảo mật mã Python để phát hiện các lỗ hổng tiềm ẩn.
   - **TruffleHog**: Tìm kiếm các khóa bí mật hoặc thông tin nhạy cảm bị lộ trong Git history.

4. **Đóng gói môi trường huấn luyện với Docker**
   4.1. Đăng nhập vào Amazon ECR bằng GitHub Action.
   4.2. Đảm bảo repository `my-app` đã tồn tại trên ECR hoặc tự động tạo mới.
   4.3. Build Docker image từ Dockerfile của đồ án và push lên Amazon ECR:

```
docker build -t $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG .
docker push $ECR_REGISTRY/$ECR_REPOSITORY:$IMAGE_TAG
```

5. **Huấn luyện mô hình với Amazon SageMaker**
   5.1. Di chuyển vào thư mục chứa mã huấn luyện (`Project/`).
   5.2. Thực thi file `train_job.py` – file này định nghĩa SageMaker training job, pipeline xử lý dữ liệu, huấn luyện và đánh giá mô hình.
   5.3. Kết quả được ghi vào file `details.txt`, sau đó kết hợp vào `report.md`.

6. **Gửi báo cáo lên Pull Request với CML**
   Nếu workflow được kích hoạt bởi Pull Request, hệ thống sẽ sử dụng **CML** để tạo comment chứa báo cáo huấn luyện lên GitHub:

```
cd Project
cml-send-comment report.md
```

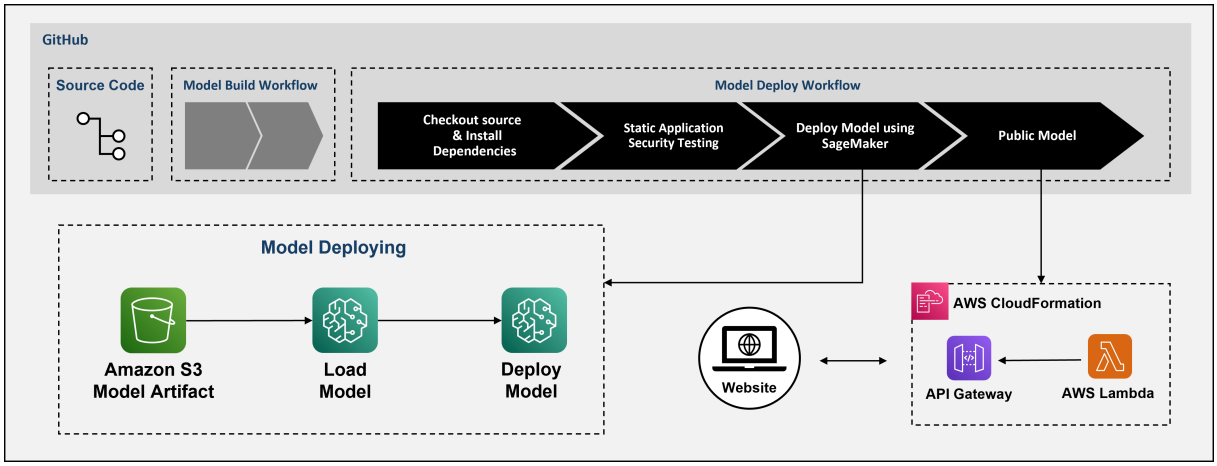
7. **Phê duyệt thủ công và triển khai mô hình**
   - Nếu mô hình đạt yêu cầu, workflow triển khai mô hình sẽ được kích hoạt thủ công (Manual Approve).
   - Phần triển khai có thể dùng thêm các công cụ như AWS Lambda, API Gateway, và SageMaker Endpoint để đưa mô hình vào sản xuất.

3.1.2 Tóm tắt quy trình

Bước	Công cụ	Mô tả
Push code	GitHub	Trigger workflow khi có Pull Request vào nhánh `dev`
Kiểm thử bảo mật	Bandit, TruffleHog	Kiểm tra an ninh, chất lượng và secrets
Đóng gói môi trường	Docker, Amazon ECR	Tạo Docker image và đẩy lên registry
Huấn luyện mô hình	SageMaker, Python	Submit training job, evaluate model
Lưu mô hình	Amazon S3	Lưu artifact sau huấn luyện
Gửi báo cáo	CML	Tạo comment kết quả lên Pull Request
Triển khai	Manual & CI/CD	Phê duyệt thủ công trước khi deploy

Bảng 3.1: Bảng thể hiện các bước, công cụ và mô tả trong quy trình CI/CD cho đồ án học máy.

3.2 Quá trình triển khai, public mô hình



Hình 3.2: Pipeline triển khai và public mô hình

Quy trình CI/CD cho đồ án triển khai mô hình học máy trên AWS SageMaker bao gồm các bước chính như sau:

Bước 1: Source Code được lưu trữ trên GitHub. Quy trình CI/CD sẽ kích hoạt khi có pull request vào nhánh `master`.

Bước 2: Workflow khởi động: GitHub Actions chạy workflow có tên `SageMaker Sentiment Analysis CI/CD - Production`.

Bước 3: Checkout mã nguồn: Dùng action `actions/checkout@v3` để lấy mã từ GitHub.

Bước 4: Cài đặt Python và thư viện phụ thuộc:

- Cài Python 3.9.
- Cài các thư viện như `boto3`, `sagemaker`, `pandas`, `scikit-learn`, `bandit`, `trufflehog`, v.v.

Bước 5: Phân tích bảo mật mã nguồn:

- **Bandit**: quét các lỗ hổng bảo mật trong mã Python.
- **TruffleHog**: tìm kiếm thông tin nhạy cảm (như secrets) bị lộ trong git history.

Bước 6: Định danh và xác thực AWS: Sử dụng `aws-actions/configure-aws-credentials@v2` để thiết lập quyền truy cập AWS (được cung cấp qua GitHub Secrets).

Bước 7: Xoá endpoint cũ (nếu có): Dùng `boto3` để xoá endpoint SageMaker nếu đã tồn tại trước đó.

Bước 8: Triển khai mô hình lên SageMaker:

- Script `deploy.py` thực hiện các bước:
  - Load mô hình từ **Amazon S3 Model Artifact**.
  - Tạo **SageMaker Endpoint**.
  - Ghi thông tin endpoint vào file `endpoint_info.json`.

Bước 9: Triển khai API Gateway và Lambda thông qua CloudFormation:

- Dùng `CloudFormation.yml` để tạo stack bao gồm:
  - API Gateway nhận request từ phía người dùng.
  - Lambda gọi tới SageMaker Endpoint tương ứng.

Bước 10: Lấy URL của API Gateway từ stack CloudFormation để hiển thị trên website.

Bước 11: Thay thế URL API vào website:

- File `index.html` có placeholder `<!-API_URL_PLACEHOLDER->` sẽ được thay bằng URL thực tế.

Bước 12: Triển khai website lên S3:

- Nếu bucket `sentiment-analysis-webapp-nhom21` chưa tồn tại, tiến hành tạo mới.
- Upload toàn bộ nội dung thư mục `Project/website/` lên bucket.

Bước 13: Hoàn tất: In ra URL truy cập website:

```
http://sentiment-analysis-webapp-nhom21.s3-website-ap-southeast-1.amazonaws.com
```

Chương 4

Demo và Mã nguồn

Chương này nhằm giới thiệu cách thức hoạt động của hệ thống thông qua một demo thực tế và cung cấp thông tin chi tiết về mã nguồn của đồ án. Mục tiêu là giúp người đọc hiểu rõ cách hệ thống vận hành và có thể tham khảo mã nguồn để tìm hiểu sâu hơn về các thành phần kỹ thuật.

Chương được chia thành hai phần chính:

- **Demo**: Mô tả chi tiết cách hệ thống hoạt động thông qua một ví dụ cụ thể.
- **Mã nguồn**: Cung cấp thông tin về cấu trúc mã nguồn, các thành phần chính và cách thức triển khai.

4.1 Demo

Để xem rõ quy trình thực hiện đồ án, thầy và các bạn có thể xem thêm tại **YouTube**.

Phần demo sẽ giới thiệu cách hệ thống phân tích cảm xúc hoạt động từ góc độ người dùng cuối, bao gồm các bước từ việc gửi yêu cầu đến nhận kết quả dự đoán. Điều này giúp minh họa rõ ràng cách các thành phần của hệ thống tương tác với nhau trong thực tế.

4.1.1 Mô tả hệ thống

Hệ thống được xây dựng để tự động hóa quá trình huấn luyện, triển khai và sử dụng mô hình phân tích cảm xúc. Các thành phần chính bao gồm:

- **Amazon SageMaker**: Huấn luyện và triển khai mô hình học máy.
- **GitHub Actions**: Tự động hóa quy trình CI/CD.
- **AWS CloudFormation**: Quản lý cơ sở hạ tầng dưới dạng mã.
- **Amazon S3**: Lưu trữ dữ liệu huấn luyện và artifact mô hình.
- **AWS Lambda và Amazon API Gateway**: Cung cấp API công khai cho ứng dụng frontend.
- **PyTorch**: Framework xây dựng mô hình LSTM.

4.1.2 Quy trình hoạt động

Quy trình hoạt động của hệ thống bao gồm các bước chính sau:

1. **Huấn luyện mô hình**: Mô hình LSTM được huấn luyện trên tập dữ liệu IMDB bằng SageMaker. Artifact của mô hình sau huấn luyện được lưu trữ trên S3.
2. **Triển khai mô hình**: Mô hình được triển khai thành một endpoint trên SageMaker thông qua GitHub Actions khi có thay đổi mã nguồn.
3. **Sử dụng API**: Người dùng gửi yêu cầu dự đoán cảm xúc thông qua API Gateway, yêu cầu này được xử lý bởi Lambda và chuyển đến endpoint của SageMaker để nhận kết quả.

4.1.3 Ví dụ cụ thể

Dưới đây là một ví dụ thực tế về cách sử dụng hệ thống để phân tích cảm xúc của một đoạn văn bản:

- **Bước 1: Chuẩn bị yêu cầu**

  Người dùng chuẩn bị một đoạn văn bản, ví dụ: "This movie is awesome!"

- **Bước 2: Gửi yêu cầu qua API**

  Yêu cầu được gửi đến API Gateway thông qua một ứng dụng frontend hoặc công cụ như Postman. API Gateway chuyển yêu cầu đến Lambda.

- **Bước 3: Xử lý yêu cầu**

  Lambda nhận yêu cầu, tiền xử lý dữ liệu (nếu cần), và gửi đến endpoint của SageMaker để dự đoán.

- **Bước 4: Nhận kết quả**

  SageMaker endpoint xử lý yêu cầu và trả về kết quả dự đoán (ví dụ: "tích cực"). Lambda nhận kết quả và gửi lại cho người dùng qua API Gateway.

- **Kết quả**: Người dùng nhận được phản hồi với dự đoán cảm xúc của đoạn văn bản.

Ví dụ này minh họa cách hệ thống hoạt động một cách liền mạch, từ việc nhận yêu cầu đến trả về kết quả, mà không yêu cầu sự can thiệp thủ công.

4.2 Mã nguồn

Mã nguồn liên quan đến đồ án đã được nhóm lưu tại **GitHub**. Tại đây cung cấp thông tin chi tiết về cấu trúc mã nguồn của đồ án, các thành phần chính và cách thức triển khai. Điều này giúp người đọc hiểu rõ cách đồ án được tổ chức và có thể tham khảo để phát triển hoặc mở rộng hệ thống.

Chương 5

Kết luận

5.1 Những kết quả đã đạt được

Trong quá trình thực hiện đồ án, nhóm đã vận dụng hiệu quả các kiến thức về DevOps để xây dựng quy trình CI/CD hoàn chỉnh cho việc triển khai mô hình Machine Learning trên nền tảng AWS. Cụ thể, nhóm đã đạt được những kết quả sau:

- Hiểu và áp dụng quy trình DevOps vào bài toán triển khai mô hình học máy, đảm bảo tính tự động, lặp lại và dễ mở rộng.
- Sử dụng thành thạo các công cụ và dịch vụ bao gồm:
  - **GitHub Actions** cho tự động hóa quy trình CI/CD.
  - **Amazon SageMaker** để triển khai và phục vụ mô hình Machine Learning.
  - **AWS CloudFormation** để quản lý cơ sở hạ tầng như API Gateway, Lambda một cách có cấu trúc.
  - **Amazon S3** để lưu trữ mô hình và triển khai website.
  - **Bandit**, **TruffleHog** để kiểm tra chất lượng và bảo mật mã nguồn.
- Triển khai thành công mô hình phân tích cảm xúc từ mã nguồn lên SageMaker, kết nối thông qua API Gateway và Lambda, đồng thời xây dựng giao diện web cho người dùng cuối.
- Đảm bảo quy trình triển khai có khả năng mở rộng và tái sử dụng, phù hợp với môi trường thực tế trong doanh nghiệp.

5.2 Những điểm còn thiếu và định hướng cải thiện

Mặc dù đã đạt được các mục tiêu đề ra, hệ thống vẫn còn một số điểm hạn chế và có thể được cải thiện trong các bước phát triển tiếp theo:

- **Bảo mật hệ thống chưa toàn diện**:
  - Các dịch vụ AWS như Lambda, S3 hay API Gateway chưa được thiết lập chính sách bảo mật chi tiết (IAM policy, CORS, JWT, v.v).

  - Chưa có lớp xác thực người dùng truy cập API và website.

- **Chưa tích hợp DataOps**: Hệ thống hiện tại chưa có cơ chế theo dõi và phản ứng khi dữ liệu thay đổi. Trong thực tế, thay đổi dữ liệu (ví dụ: dữ liệu huấn luyện mới) cần kích hoạt lại pipeline để:

  - Tiền xử lý lại dữ liệu.
  - Huấn luyện lại mô hình.
  - Triển khai mô hình mới (A/B testing, rollback).

  Tích hợp DataOps sẽ giúp đảm bảo mô hình luôn cập nhật và thích nghi với dữ liệu mới theo hướng tự động hóa.

- **Chưa có hệ thống giám sát (monitoring)**:

  - Chưa thiết lập log, metric và cảnh báo (alerting) khi mô hình gặp lỗi hoặc hoạt động bất thường.
  - Có thể tích hợp thêm CloudWatch hoặc Prometheus + Grafana để theo dõi sức khỏe hệ thống.

- **Thiếu kiểm thử mô hình (Model Validation)**: Trong pipeline hiện tại chưa có bước kiểm thử chất lượng mô hình (accuracy, F1-score, drift detection) trước khi triển khai lên môi trường production.

- **Khả năng mở rộng giao diện người dùng**: Giao diện web còn đơn giản, có thể phát triển thành hệ thống dashboard tương tác, hỗ trợ nhập dữ liệu hàng loạt hoặc phân tích lịch sử truy vấn.

5.3 Định hướng phát triển trong tương lai

Dựa trên các phân tích trên, nhóm đề xuất một số hướng phát triển:

- Tích hợp CI/CD với DataOps và MLOps toàn diện.
- Bổ sung bước đánh giá mô hình tự động sau mỗi lần huấn luyện.
- Tăng cường bảo mật ứng dụng và API theo chuẩn công nghiệp.
- Tích hợp hệ thống giám sát hoạt động và cảnh báo tự động.
- Mở rộng khả năng tương tác người dùng thông qua frontend hiện đại (React/Vue).

Tóm lại, đồ án đã giúp nhóm củng cố và mở rộng hiểu biết về DevOps trong bối cảnh triển khai hệ thống học máy thực tế, đồng thời tạo tiền đề vững chắc cho việc phát triển các đồ án AI quy mô lớn và chuyên nghiệp hơn trong tương lai.

TÀI LIỆU THAM KHẢO

[1] *AWS Documentation*. Amazon Web Services. 2024. URL: https://docs.aws.amazon.com/.

[2] *GitHub Actions Documentation*. GitHub. 2024. URL: https://docs.github.com/en/actions.

[3] AWS Samples. *MLOps example using Amazon SageMaker Pipeline and GitHub Actions*. GitHub. 2023. URL: https://github.com/aws-samples/mlops-sagemaker-github-actions.

[4] Haythem Tellili. *Machine Learning CI/CD Pipeline with Github Actions and Amazon SageMaker*. Medium. 2023. URL: https://medium.com/@haythemtellili/machine-learning-ci-cd-pipeline-with-github-actions-and-amazon-sagemaker-f99818b7506a.