

ĐẠI HỌC KINH TẾ TP.HỒ CHÍ MINH (UEH)
TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ
KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH
BỘ MÔN CÔNG NGHỆ THÔNG TIN



**BÁO CÁO ĐÒ ÁN
TÍNH TOÁN HIỆU SUẤT CAO
(HPC - High Performance Computing)**

Đề tài:

Triển khai Python trên Docker Image Official

GVHD:
Mã học phần:
Nhóm thực hiện:
Thành viên nhóm:

TS.GVC Nguyễn Quốc Hùng
24D1INF50907702

5

Lê Trần Khánh Phú (Trưởng nhóm)
Huỳnh Nguyễn Anh Cường
Huỳnh Trịnh Tiến Khoa
Trương Thanh Phong
Nguyễn Ngọc Tường Vy

TP. Hồ Chí Minh, Tháng 5/2024

MỤC LỤC

| | |
|--|----|
| DANH MỤC HÌNH ẢNH. | 3 |
| DANH MỤC BẢNG BIỂU. | 5 |
| DANH MỤC TỪ VIẾT TẮT. | 6 |
| Lời mở đầu. | 7 |
| BẢNG PHÂN CÔNG CÁC THÀNH VIÊN | 8 |
| Chương 1. GIỚI THIỆU TÍNH TOÁN HIỆU SUẤT CAO (HPC) VÀ DỰ ÁN THỰC HIỆN | 9 |
| 1.1 GIỚI THIỆU VỀ TÍNH TOÁN HIỆU SUẤT CAO (HPC) | 9 |
| 1.2 TỔNG QUAN ĐỀ TÀI | 11 |
| Chương 2. CƠ SỞ LÝ THUYẾT VỀ CHỦ ĐỀ NGHIÊN CỨU | 14 |
| 2.1 DOCKER | 14 |
| 2.2 PYTHON | 16 |
| Chương 3. TRIỂN KHAI THỬ NGHIỆM HỆ THỐNG | 22 |
| 3.1 ĐỀ XUẤT MÔ HÌNH TRIỂN KHAI | 22 |
| 3.2 XÂY DỰNG ỨNG DỤNG PYTHON | 25 |
| 3.3 TRIỂN KHAI THỬ NGHIỆM HỆ THỐNG | 40 |
| Chương 4. BÀN LUẬN VÀ ĐÁNH GIÁ | 63 |
| 4.1 BÀN LUẬN VÀ ĐÁNH GIÁ TRIỂN KHAI PYTHON TRÊN DOCKER IMAGE OFFICIAL | 63 |
| 4.2 BÀN LUẬN VÀ ĐÁNH GIÁ TÍNH NĂNG TRUY CẬP VÀO LOCAL URL VÀ PUBLIC URL | 63 |
| KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN | 65 |
| TÀI LIỆU THAM KHẢO | 67 |

DANH MỤC HÌNH ẢNH.

| | |
|---|----|
| Hình 1.1-1 Mô hình triển khai hệ thống HPC | 9 |
| Hình 2.1-1 Logo Docker..... | 14 |
| Hình 2.1-2 Kiến trúc của Docker | 15 |
| Hình 3.1-1 Phương hướng xây dựng ứng dụng Python | 23 |
| Hình 3.2-1 Bộ dữ liệu UIT-ViSFD..... | 25 |
| Hình 3.2-2 Tham số mô hình Naive Bayes | 31 |
| Hình 3.2-3 Tham số mô hình Logistic Regression..... | 33 |
| Hình 3.2-4 Bộ dữ liệu UIT-ViSFD khi giảm mẫu | 34 |
| Hình 3.2-5 Bộ dữ liệu UIT-ViSFD khi tăng mẫu..... | 36 |
| Hình 3.2-6 Ma trận nhầm lẫn của mô hình Logistic Regression..... | 37 |
| Hình 3.2-7 Ma trận nhầm lẫn của mô hình Naive Bayes | 37 |
| Hình 3.2-8 Giao diện của Web | 39 |
| Hình 3.2-9 Chạy thử đánh giá "Máy chậm, lag, pin yếu" | 40 |
| Hình 3.3-1 Thư mục đồ án được tạo..... | 40 |
| Hình 3.3-2 Cấu trúc file python_app..... | 45 |
| Hình 3.3-3 Khởi tạo image pythonapp thành công | 48 |
| Hình 3.3-4 Kiểm tra các Image trong Windows PowerShell | 49 |
| Hình 3.3-5 Kiểm tra các Image trong Docker | 49 |
| Hình 3.3-6 Khởi chạy Container sử dụng Image pythonapp..... | 50 |
| Hình 3.3-7 Các đường link truy cập web | 50 |
| Hình 3.3-8 Kiểm tra các Container đang chạy trong Windows PowerShell | 50 |
| Hình 3.3-9 Kiểm tra các Container đang chạy trong Docker | 50 |
| Hình 3.3-10 Giao diện Web khi truy cập Local URL | 51 |
| Hình 3.3-11 Chạy thử nghiệm với một bình luận toàn khen | 51 |
| Hình 3.3-12 Chạy thử nghiệm với một bình luận vừa khen vừa chê | 52 |
| Hình 3.3-13 Chạy thử nghiệm với bình luận toàn chê | 52 |
| Hình 3.3-14 Giao diện web khi truy cập public URL | 53 |
| Hình 3.3-15 Chạy thử nghiệm với bình luận toàn khen | 53 |
| Hình 3.3-16 Chạy thử nghiệm với bình luận vừa khen vừa chê | 54 |
| Hình 3.3-17 Chạy thử nghiệm với bình luận toàn chê | 54 |
| Hình 3.3-18 Kiểm tra các Container đang chạy trong Windows PowerShell | 55 |
| Hình 3.3-19 Kiểm tra các Container đang chạy trong Docker | 55 |
| Hình 3.3-20 Không truy cập được Local URL..... | 55 |
| Hình 3.3-21 Không truy cập được Public URL..... | 56 |
| Hình 3.3-22 Kiểm tra các Container đang chạy trong Windows PowerShell | 56 |
| Hình 3.3-23 Kiểm tra các Container đang chạy trong Docker | 57 |
| Hình 3.3-24 Truy cập được Local URL | 57 |
| Hình 3.3-25 Không truy cập được Public URL..... | 58 |
| Hình 3.3-26 Thông tin Container khi chạy lại..... | 59 |
| Hình 3.3-27 Truy cập được Public URL | 59 |
| Hình 3.3-28 Hồ sơ người dùng trên Docker | 60 |
| Hình 3.3-29 Màn hình khi login Docker thành công | 60 |
| Hình 3.3-30 Kiểm tra các Image trong Windows PowerShell | 60 |
| Hình 3.3-31 Đẩy Image lên Docker Hub..... | 61 |
| Hình 3.3-32 Image pythonapp đã được đưa lên Docker Hub | 62 |
| Hình 3.3-33 Tải image pythonapp về | 62 |

Hình 3.3-34 Kiểm tra các Image trong Windows PowerShell 62

DANH MỤC BẢNG BIỂU.

| | |
|---|----|
| Bảng 3.2-1 Tổng quát bộ dữ liệu UIT-ViSFD | 25 |
| Bảng 3.2-2 Tổng quát bộ dữ liệu UIT-ViSFD sau khi xóa cột | 25 |

DANH MỤC TỪ VIẾT TẮT.

| STT | Ký hiệu viết tắt | Chữ viết đầy đủ | Ý nghĩa |
|-----|------------------|-----------------------------|--|
| 1 | HPC | High Performance Computing | Tính toán hiệu suất cao |
| 2 | VS Code | Visual Studio Code | Trình soạn thảo mã nguồn được phát triển bởi Microsoft |
| 3 | NLP | Natural Language Processing | Xử lý ngôn ngữ tự nhiên |

Lời mở đầu.

Để hoàn thành đề tài này, trước tiên nhóm chúng em xin gửi đến khoa Công nghệ thông tin, trường Công nghệ và thiết kế - UEH lời cảm ơn chân thành và sâu sắc nhất vì đã đưa môn Tính toán hiệu suất cao vào chương trình giảng dạy, giúp chúng em được học, khai thác và thực hành nhiều kiến thức bổ ích từ học phần này.

Đặc biệt, nhóm chúng em xin gửi lời cảm ơn sâu sắc nhất đến giảng viên giảng dạy TS. Nguyễn Quốc Hùng đã nhiệt tình giảng dạy, truyền tải, hướng dẫn chúng em nhiều kiến thức bổ ích trong suốt thời gian học. Cảm ơn sự hỗ trợ, giúp đỡ tận tình của Thầy để nhóm chúng em có thể thực hiện đề tài một cách tốt nhất.

Vì kiến thức của nhóm còn nhiều hạn chế, trong quá trình học và hoàn thiện đề tài chắc chắn chúng em không tránh khỏi những sai sót. Nhóm chúng em rất mong nhận được nhận xét, ý kiến và đánh giá từ Thầy để đề tài của nhóm được tốt hơn.

Nhóm chúng em xin chân thành cảm ơn Thầy. Kính chúc Thầy có nhiều sức khỏe, thành công và hạnh phúc.

BẢNG PHÂN CÔNG CÁC THÀNH VIÊN

| TT | Họ và tên | Công việc phụ trách | Mức độ hoàn thành |
|----|---|--|-------------------|
| 1. | Lê Trần Khánh Phú <i>(Trưởng nhóm)</i> | Viết lời cảm ơn Xây dựng ứng dụng Python Triển khai ứng dụng Python trên Docker Image Official Triển khai Image lên nền tảng Docker Hub Thiết kế slide trình bày | 100% |
| 2. | Huỳnh Nguyễn Anh Cường | Tiền xử lý dữ liệu Đánh giá mô hình Viết kết luận và đề xuất hướng phát triển Thiết kế slide trình bày | 100% |
| 3. | Huỳnh Trịnh Tiến Khoa | Xây dựng mô hình Naive Bayes Bàn luận và đánh giá Tổng hợp file dự án Thiết kế slide trình bày | 100% |
| 4. | Trương Thanh Phong | Cơ sở lý thuyết Docker Cơ sở lý thuyết Python Xây dựng Web UI Thiết kế slide trình bày | 100% |
| 5. | Nguyễn Ngọc Tường Vy | Giới thiệu về Tính Toán Hiệu Suất Cao (HPC) Tổng Quan Đề Tài Đề xuất mô hình và phương hướng triển khai Xây dựng mô hình Maxent Thiết kế slide trình bày | 100% |

Chương 1. GIỚI THIỆU TÍNH TOÁN HIỆU SUẤT CAO (HPC) VÀ DỰ ÁN THỰC HIỆN

1.1 GIỚI THIỆU VỀ TÍNH TOÁN HIỆU SUẤT CAO (HPC)

1.1.1 Hệ thống HPC là gì?

HPC là viết tắt của High Performance Computing, hay còn gọi là **siêu máy tính** hoặc **máy tính hiệu năng cao**. Đây là thuật ngữ dùng để chỉ các hệ thống máy tính có khả năng xử lý dữ liệu và thực hiện các phép tính phức tạp với tốc độ cao hơn rất nhiều so với máy tính thông thường.

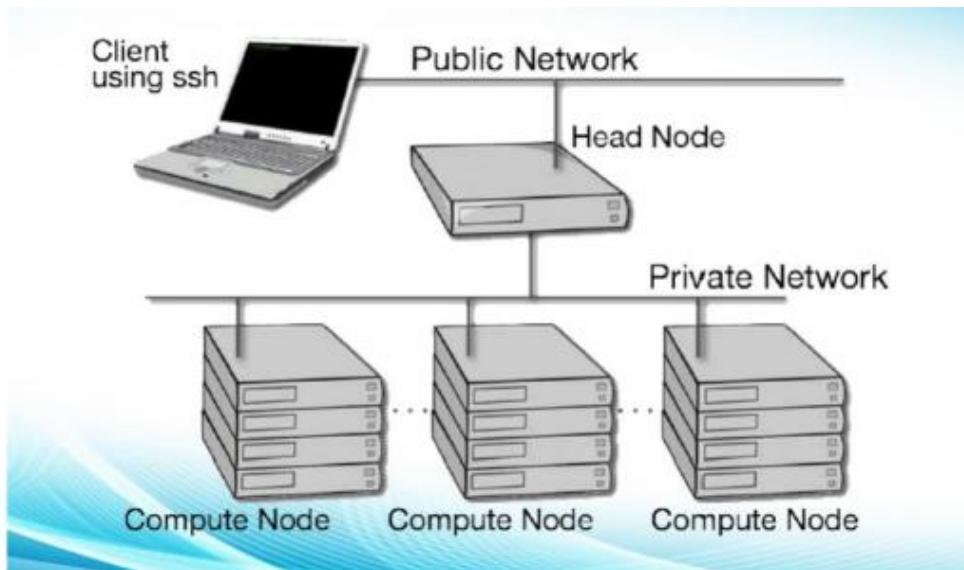
- **Đặc điểm của hệ thống HPC:**

- **Khả năng xử lý mạnh mẽ:** Có thể thực hiện hàng tỷ phép tính mỗi giây (flops), cao hơn hàng nghìn lần so với máy tính cá nhân. Nhờ đó, HPC có thể giải quyết các bài toán phức tạp trong thời gian ngắn hơn rất nhiều so với máy tính thông thường.
- **Khả năng mở rộng:** HPC có thể được cấu hình với nhiều nút (node) để đáp ứng nhu cầu xử lý ngày càng tăng.
- **Độ tin cậy cao:** HPC được thiết kế để hoạt động liên tục trong thời gian dài với độ tin cậy cao.
- **Khả năng lập trình:** HPC có thể được lập trình bằng các ngôn ngữ lập trình chuyên dụng như C, Fortran và Python...

- **Thành phần:**

- Phần cứng :
 - Cơ bản: CPU, RAM, HDD, Card Network...
 - Mở rộng: Card tăng tốc đồ họa GPU (tích hợp hàng ngàn core); Bộ đồng xử lý(Co-processor).
- Phần mềm:
 - Kiểm soát và tính toán năng lực từ các Node trong Cluster.
 - Quản lý các tài khoản truy cập.

1.1.2 Mô hình triển khai hệ thống HPC



Hình 1.1-1 Mô hình triển khai hệ thống HPC

“ Các máy tính được kết nối với nhau qua mạng nội bộ (Private Network) và kết nối với một máy tính điều khiển (Head Node) hay gọi là nút điều khiển, nút này kết nối ra ngoài bằng mạng công khai (Public Network) qua giao thức bảo mật SSH. Mỗi máy tính tham gia trong hệ thống HPC được định nghĩa là một nút tính toán, một hệ thống HPC gồm nhiều nút tính toán và có thể bổ sung thêm nhiều nút tính toán nếu cần thiết nhằm nâng cao hiệu năng hệ thống. ”

Mô hình kết cấu hệ thống HPC với mạng nội bộ, nút điều khiển, mạng công khai và nút tính toán mang lại nhiều lợi ích cho người dùng, bao gồm hiệu suất cao, khả năng mở rộng, độ tin cậy cao và tính linh hoạt. Hệ thống HPC đóng vai trò quan trọng trong việc thúc đẩy sự phát triển của khoa học và công nghệ.

1.1.3 Kiến trúc của hệ thống HPC

Hệ thống HPC (High Performance Computing) thường được thiết kế theo hai kiến trúc xử lý chính: **MPP (Massively Parallel Processing)** và **CC (Cluster Computing)**. Mỗi kiến trúc có những ưu điểm và nhược điểm riêng, phù hợp với những nhu cầu sử dụng khác nhau.

1.1.3.1 Kiến trúc MPP (Massively Parallel Processing)

Hệ thống MPP bao gồm một tập hợp lớn các máy tính độc lập được kết nối với nhau thông qua mạng tốc độ cao. Mỗi máy tính trong hệ thống MPP có bộ nhớ và CPU riêng. Các tác vụ tính toán được chia nhỏ thành nhiều phần nhỏ và được phân phối cho các máy tính khác nhau trong hệ thống để thực hiện đồng thời.

- **Ưu điểm:**
 - Khả năng mở rộng cao: Hệ thống MPP có thể được mở rộng dễ dàng bằng cách bổ sung thêm các máy tính mới.
 - Hiệu suất cao: Hệ thống MPP có thể thực hiện các phép tính với tốc độ cao nhờ sự kết hợp của nhiều máy tính.
 - Khả năng chịu lỗi cao: Nếu một máy tính trong hệ thống MPP bị lỗi, các máy tính khác có thể tiếp tục hoạt động mà không bị ảnh hưởng.
- **Nhược điểm:**
 - Chi phí cao: Hệ thống MPP có thể có chi phí ban đầu cao do cần mua nhiều máy tính.
 - Độ phức tạp cao: Hệ thống MPP có thể phức tạp hơn để quản lý và vận hành so với các hệ thống HPC khác.

Ví dụ: Hệ thống Cray XC50 là một hệ thống MPP được sử dụng cho nghiên cứu khoa học và kỹ thuật. Hệ thống IBM Power System 9 là một hệ thống MPP được sử dụng cho các ứng dụng kinh doanh.

1.1.3.2 Kiến trúc CC (Cluster Computing)

Hệ thống CC bao gồm một tập hợp các máy tính được kết nối với nhau thông qua mạng cục bộ. Các máy tính trong hệ thống CC có thể chia sẻ tài nguyên như bộ nhớ và CPU. Các tác vụ tính toán có thể được thực hiện trên một hoặc nhiều máy tính trong hệ thống CC.

- **Ưu điểm:**

- Chi phí thấp: Hệ thống CC có thể có chi phí ban đầu thấp hơn so với hệ thống MPP.
- Dễ dàng triển khai: Hệ thống CC có thể dễ dàng triển khai và quản lý hơn so với hệ thống MPP.
- **Nhược điểm:**
 - Khả năng mở rộng hạn chế: Hệ thống CC có thể khó mở rộng hơn so với hệ thống MPP.
 - Hiệu suất thấp hơn: Hệ thống CC có thể có hiệu suất thấp hơn so với hệ thống MPP do tài nguyên được chia sẻ giữa các máy tính.

Ví dụ: Hệ thống Beowulf là một hệ thống CC miễn phí và mã nguồn mở. Hệ thống HTCondor là một hệ thống CC được sử dụng để phân phối các tác vụ tính toán trên nhiều máy tính.

1.1.4 Ứng dụng và hướng phát triển

Hệ thống HPC được ứng dụng trong nhiều lĩnh vực quan trọng như:

- **Khoa học:** HPC được sử dụng trong nhiều lĩnh vực khoa học như vật lý, hóa học, sinh học, y học, khoa học Trái đất, v.v. để mô phỏng các hiện tượng phức tạp và giải quyết các bài toán khoa học.
- **Kỹ thuật:** HPC được sử dụng trong kỹ thuật để thiết kế sản phẩm mới, tối ưu hóa quy trình sản xuất và phân tích dữ liệu kỹ thuật.
- **Tài chính:** HPC được sử dụng trong lĩnh vực tài chính để phân tích thị trường, quản lý rủi ro và phát triển các mô hình tài chính.
- **Quân sự:** HPC được sử dụng trong quân sự để mô phỏng chiến tranh, phát triển vũ khí và hệ thống phòng thủ.

Xu hướng phát triển ngày nay của hệ thống HPC:

- **HPC đám mây:** HPC đang ngày càng được triển khai trên nền tảng đám mây, giúp người dùng có thể truy cập và sử dụng HPC dễ dàng hơn.
- **HPC trí tuệ nhân tạo:** HPC đang được kết hợp với trí tuệ nhân tạo (AI) để phát triển các ứng dụng mới trong nhiều lĩnh vực.
- **HPC lượng tử:** HPC lượng tử là một lĩnh vực nghiên cứu mới nổi, hứa hẹn sẽ mang lại hiệu suất xử lý cao hơn gấp nhiều lần so với HPC truyền thống.

1.2 TỔNG QUAN ĐỀ TÀI

1.2.1 Xác định đề tài

Đề tài “**Triển khai Python trên Docker Image Official**” thực hiện sử dụng Python tạo công cụ dự đoán đánh giá sản phẩm điện thoại từ bình luận người dùng trên các sàn thương mại điện tử trên Docker. Đề tài nhấn mạnh việc triển khai một ứng dụng NLP trên Docker, đòi hỏi sự kết hợp giữa việc xây dựng một môi trường chạy ứng dụng hiệu quả và triển khai một ứng dụng có tính toán phức tạp.

1.2.2 Lý do lựa chọn đề tài

Trong quá trình phát triển phần mềm, việc thiết lập môi trường làm việc phù hợp cho dự án là một thách thức không nhỏ. Thực tế, nhiều lập trình viên đã phải dành hàng giờ, thậm chí nhiều ngày, để cài đặt các thư viện cần thiết cho dự án trên máy tính cá nhân. Thông thường, khi tham gia vào một dự án phát triển phần mềm, lập trình viên sẽ tải mã nguồn của dự án về, sau đó thiết lập môi trường và kiểm tra mã nguồn trên máy tính cá nhân trước khi tiến hành phát triển thêm các tính năng mới.

Đối với các dự án sử dụng ngôn ngữ lập trình Python, các thư viện cần thiết thường được khai báo trong tệp `requirements.txt`. Một vấn đề phổ biến là sự không tương thích giữa các phiên bản thư viện và nền tảng máy tính của lập trình viên. Điều này dễ xảy ra khi phiên bản thư viện yêu cầu đã quá cũ hoặc máy tính sử dụng các công nghệ mới hơn (ví dụ như MacBook với chip M1 hay M2). Để khắc phục vấn đề này, đôi khi cần phải build lại thư viện từ mã nguồn, quá trình này thường gặp lỗi do thiếu các thư viện phụ thuộc hoặc việc liên kết các thư viện không đúng cách.

Để giải quyết vấn đề này và giảm thiểu sự phụ thuộc vào nền tảng phần cứng, Docker là một giải pháp tối ưu. Docker cho phép đóng gói ứng dụng vào các container, đảm bảo cung cấp một môi trường đồng nhất. Điều này không chỉ giúp lập trình viên dễ dàng thiết lập và chạy ứng dụng trên bất kỳ máy tính nào mà còn tăng tính linh hoạt và khả năng triển khai ứng dụng một cách nhât quán trên nhiều môi trường khác nhau.

Đồng thời, trong bối cảnh mua sắm trực tuyến ngày càng phát triển, các bình luận và đánh giá của người dùng về sản phẩm trở thành nguồn thông tin quan trọng, giúp người tiêu dùng đưa ra quyết định mua hàng thông minh hơn. Đối với các nhà bán lẻ và nhà sản xuất, việc phân tích các bình luận này giúp họ hiểu rõ hơn về phản hồi của khách hàng, từ đó cải thiện sản phẩm và dịch vụ. Tuy nhiên, việc phân tích thủ công các bình luận là không khả thi do số lượng lớn và sự đa dạng của ngôn ngữ sử dụng.

Vì vậy, việc tạo ra một công cụ tự động dự đoán đánh giá sản phẩm điện thoại từ bình luận người dùng trên các sàn thương mại điện tử là rất cần thiết. Công cụ này không chỉ giúp người tiêu dùng dễ dàng tổng hợp và đánh giá thông tin sản phẩm mà còn cung cấp cho các nhà bán lẻ và nhà sản xuất cái nhìn sâu sắc về phản hồi của khách hàng, giúp họ cải thiện chất lượng sản phẩm và dịch vụ.

Kết hợp hai yếu tố trên, nhóm chọn đề tài **“Triển khai Python trên Docker Image Official”** để tạo công cụ dự đoán đánh giá sản phẩm điện thoại từ bình luận người dùng trên các sàn thương mại điện tử, không những chỉ giải quyết các vấn đề thực tiễn trong quá trình phát triển phần mềm mà còn tận dụng được những lợi ích vượt trội của Docker trong việc quản lý môi trường và triển khai ứng dụng.

1.2.3 Mục tiêu nghiên cứu

Mục tiêu cuối cùng là tạo ra một công cụ dự đoán đánh giá sản phẩm điện thoại được viết bằng ngôn ngữ Python trên Docker. Hy vọng rằng nghiên cứu này sẽ giúp đơn giản hóa quy trình triển khai và quản lý môi trường cho ứng dụng Python, đồng thời cung cấp một giải pháp linh hoạt và dễ dàng tái sử dụng, giúp đảm bảo tính nhất quán và tin cậy trong quá trình chạy ứng dụng trên các nền tảng khác nhau. Bên cạnh đó, nghiên cứu này sẽ đóng góp vào việc hiểu sâu hơn về cách người dùng tương tác với sản phẩm trên sàn thương mại điện tử và cung cấp một cơ sở cho việc phát triển chiến lược kinh doanh dựa trên ý kiến của khách hàng.

1.2.4 Phương pháp nghiên cứu

Phương pháp nghiên cứu tập trung vào việc sử dụng Docker để xây dựng và triển khai môi trường chạy ứng dụng Python. Điều này bao gồm việc sử dụng một Dockerfile để định nghĩa môi trường chạy, sử dụng các Docker image chính thức của Python và đóng gói ứng dụng vào các container để đảm bảo cung cấp một môi trường nhất quán, giúp dễ dàng chạy và triển khai ứng dụng trên bất kỳ máy tính nào.

1.2.5 Tài nguyên sử dụng

- Ngôn ngữ lập trình: Python.
- Docker image chính thức của Python.
- Các công cụ và tài liệu hỗ trợ từ Docker, như Dockerfile và Docker Hub.
- Tài nguyên mạng và hướng dẫn từ cộng đồng Docker và Python để nắm vững quy trình triển khai và quản lý ứng dụng trên Docker.
- Bộ dữ liệu về phản hồi đối với sản phẩm điện thoại thông minh của người Việt-UIT-ViSFD.
- Danh sách teencode được lấy từ github tác giả Nguyễn Văn Hiếu để hỗ trợ cho việc biên dịch các bình luận: [teencode](#)
- Danh sách stopword, nguồn: [vnstopword.txt | Powered by Box](#)

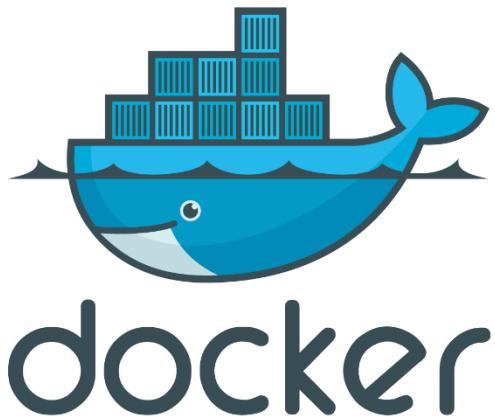
Chương 2. CƠ SỞ LÝ THUYẾT VỀ CHỦ ĐỀ NGHIÊN CỨU

2.1 DOCKER

2.1.1 Khái niệm

Docker là một nền tảng mã nguồn mở để xây dựng, đóng gói và triển khai ứng dụng. Nó cho phép tạo các môi trường độc lập và tách biệt để khởi chạy và phát triển ứng dụng và môi trường này được gọi là nền tảng ảo hóa giúp ảo hóa ứng dụng bằng cách đóng gói ứng dụng cùng với tất cả các phụ thuộc của nó, bao gồm cả mã nguồn, thư viện hệ thống, cài đặt và các tệp khác, thành một đơn vị độc lập gọi là container.

Container Docker được cô lập với nhau và với hệ thống máy chủ, cho phép chúng chạy mà không ảnh hưởng lẫn nhau hoặc với hệ thống cơ bản. Container Docker được thiết kế để chạy nhất quán trên nhiều môi trường, từ máy tính xách tay đến các cụm máy chủ lớn và dễ dàng chia sẻ ứng dụng của người dùng với người khác bằng cách chia sẻ container hoặc image Docker.

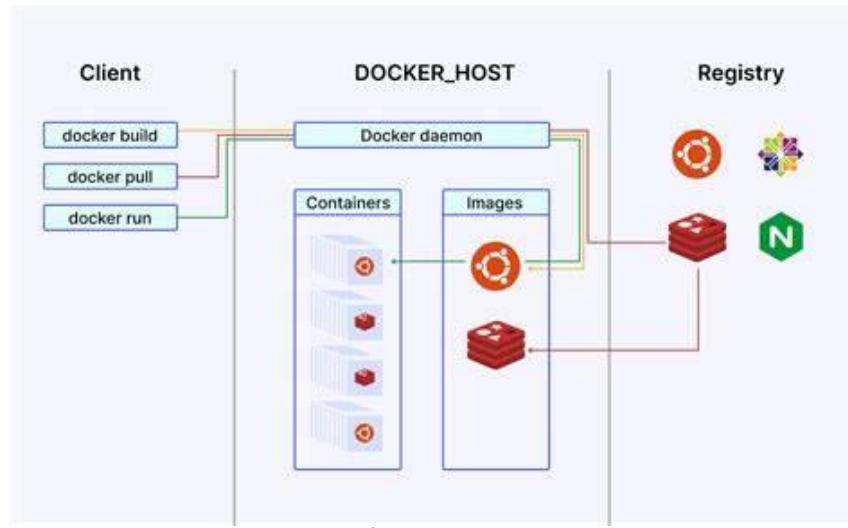


Hình 2.1-1 Logo Docker

2.1.2 Cách thức hoạt động Docker

Docker hoạt động bằng cách sử dụng công nghệ ảo hóa nhẹ được gọi là containerization. Containerization là một quy trình đóng gói phần mềm và tất cả các phụ thuộc của nó vào một gói duy nhất, có thể dễ dàng triển khai và chạy trên nhiều môi trường. Hiểu đơn giản rằng containerization cho phép Docker tạo ra một môi trường thực thi riêng biệt cho mỗi ứng dụng, bao gồm hệ thống tệp, thư viện và các cài đặt khác. Điều này giúp cô lập các ứng dụng với nhau và với hệ thống máy chủ.

Docker sử dụng các image để lưu trữ các bản sao tĩnh của các ứng dụng. Image Docker là một tập tin chỉ đọc chứa tất cả các tệp cần thiết để chạy một ứng dụng. Khi bạn muốn chạy một ứng dụng, Docker sẽ tạo một container từ image. Container là một phiên bản chạy của image. Nó bao gồm tất cả các tệp và tài nguyên cần thiết để chạy ứng dụng. Và Docker cũng cung cấp một registry để lưu trữ và chia sẻ các image. Registry là một kho lưu trữ tập trung cho các image Docker. Bạn có thể tải image từ registry hoặc đẩy image của riêng bạn lên registry.



Hình 2.1-2 Kiến trúc của Docker

Để tạo được image Docker thì cần phải có Dockerfile, là tập tin văn bản chứa các hướng dẫn để xây dựng image. Docker Engine sử dụng Dockerfile để thực thi các hướng dẫn và tạo image Docker.

Trong đó, cấu trúc của một Docker file như sau:

```

FROM <image gốc>
WORKDIR <thư mục làm việc>
COPY <tệp nguồn> <đường dẫn đích>
RUN <lệnh>
CMD <lệnh>

```

Chi tiết từng phần như sau:

- FROM: Chỉ định image gốc để xây dựng image mới.
- WORKDIR: Thiết lập thư mục làm việc mặc định cho image mới.
- COPY: Sao chép tệp từ máy chủ cục bộ vào image mới.
- RUN: Thực thi các lệnh trong image mới.
- CMD: Chỉ định lệnh khởi động cho ứng dụng khi chạy image

Như vậy, Docker hoạt động bằng cách sử dụng Dockerfile để định nghĩa cách xây dựng một image. Dockerfile chứa các lệnh và chỉ dẫn cần thiết để cài đặt và thiết lập môi trường cho ứng dụng. Docker Engine, phần mềm nền tảng, đọc Dockerfile và tạo ra một image - một bản mẫu bất biến của môi trường ứng dụng. Image này sau đó có thể được lưu trữ trong Docker Registry, nơi chứa các image sẵn sàng để sử dụng hoặc chia sẻ. Khi cần chạy ứng dụng, Docker Engine sẽ tạo ra một hoặc nhiều container từ image; container là các instance chạy của image, cung cấp môi trường cô lập cho ứng dụng. Mỗi container hoạt động độc lập và có thể được khởi động, dừng hoặc xóa mà không ảnh hưởng đến hệ thống hoặc các container khác.

2.1.3 Lý do tại sao nên sử dụng Docker

Có nhiều lý do khiến Docker trở thành một công cụ có giá trị cho các ứng dụng HPC (High Performance Computing):

- **Cách ly và tính di động:** Docker cung cấp cách ly cho các ứng dụng HPC, cho phép chúng chạy mà không ảnh hưởng lẫn nhau hoặc với hệ thống cơ bản. Điều này giúp cải thiện độ ổn định và bảo mật, đồng thời đơn giản hóa việc gỡ lỗi. Hơn nữa, Docker images có thể được dễ dàng di chuyển giữa các hệ thống HPC, giúp đơn giản hóa việc triển khai và quản lý ứng dụng.
- **Tính tái tạo:** Docker đảm bảo rằng các ứng dụng HPC luôn chạy theo cùng một cách, bất kể môi trường nào. Điều này đạt được bằng cách đóng gói ứng dụng cùng với tất cả các phụ thuộc của nó trong một image Docker. Khi image được triển khai, nó sẽ tạo ra một môi trường thực thi riêng biệt, đảm bảo rằng ứng dụng sẽ chạy chính xác như mong muốn.
- **Hiệu quả:** Docker có thể giúp cải thiện hiệu quả sử dụng tài nguyên trong hệ thống HPC. Bằng cách đóng gói nhiều ứng dụng vào một container, Docker có thể giảm thiểu overhead liên quan đến việc chạy nhiều quy trình riêng biệt. Ngoài ra, Docker có thể sử dụng các tính năng như image layering để tối ưu hóa việc sử dụng dung lượng đĩa.
- **Quản lý dễ dàng:** Docker cung cấp một cách thức đơn giản để quản lý các ứng dụng HPC. Các image Docker có thể được dễ dàng lưu trữ, phiên bản và triển khai bằng các công cụ dòng lệnh hoặc API. Điều này giúp đơn giản hóa việc quản lý vòng đời của ứng dụng và cho phép các nhà khoa học tập trung vào công việc của họ thay vì lo lắng về cơ sở hạ tầng.
- **Hỗ trợ cho nhiều ngôn ngữ và thư viện:** Docker hỗ trợ nhiều ngôn ngữ lập trình và thư viện phổ biến, bao gồm Python, R, C ++ và Java. Điều này làm cho nó trở thành một công cụ linh hoạt cho nhiều loại ứng dụng HPC.
- **Cộng đồng lớn và phát triển nhanh chóng:** Docker có một cộng đồng lớn và phát triển nhanh chóng đang tích cực đóng góp vào sự phát triển của dự án. Điều này đảm bảo rằng Docker sẽ tiếp tục được cải thiện và cập nhật các tính năng mới.

Nhìn chung, Docker là một công cụ mạnh mẽ có thể mang lại nhiều lợi ích cho các tổ chức và cá nhân. Nó giúp đơn giản hóa việc phát triển, triển khai và quản lý ứng dụng, đồng thời tiết kiệm thời gian, tiền bạc và tài nguyên. Ngoài những lý do được liệt kê ở trên, Docker còn có thể mang lại nhiều lợi ích khác, chẳng hạn như Docker giúp các nhà phát triển dễ dàng chia sẻ ứng dụng của họ với nhau, tăng tốc độ phát triển ứng dụng bằng cách cung cấp một cách nhanh chóng và dễ dàng để thử nghiệm các thay đổi và giảm chi phí vận hành ứng dụng bằng cách cải thiện hiệu quả sử dụng tài nguyên. Cùng với cú pháp đơn giản và dễ hiểu của Docker cung cấp cho ta toàn quyền kiểm soát. Việc áp dụng rộng rãi nền tảng này đã tạo ra một hệ sinh thái bền vững gồm các công cụ và ứng dụng sẵn sàng sử dụng và hoạt động với Docker.

2.2 PYTHON

2.2.1 Giới thiệu

Python, được phát triển bởi Guido van Rossum vào năm 1989, là ngôn ngữ lập trình bậc cao nổi tiếng với cú pháp đơn giản, dễ học, dễ sử dụng, linh hoạt và đa dạng thư viện. Nhờ những ưu điểm này, Python đã trở thành một trong những ngôn ngữ lập trình phổ biến nhất hiện nay, được ứng dụng rộng rãi trong nhiều lĩnh vực bao gồm Docker và HPC. Docker giúp triển khai và quản lý ứng dụng Python hiệu quả, trong khi HPC giúp tăng tốc độ xử lý cho các ứng dụng Python đòi hỏi tính toán mạnh mẽ.

2.2.2 Tại sao nên lựa chọn Docker làm nơi triển khai Python?

Sử dụng Docker để triển khai Python mang lại nhiều lợi ích so với các phương pháp truyền thống. Đầu tiên phương pháp này có tính cách ly và tính nhất quán khi mà Docker tạo ra các container riêng biệt cho mỗi ứng dụng, giúp cô lập ứng dụng khỏi môi trường hệ thống và các ứng dụng khác. Điều này đảm bảo rằng ứng dụng Python luôn chạy với cùng cấu hình, bất kể môi trường triển khai nào. Tính nhất quán này rất quan trọng cho việc phát triển, thử nghiệm và triển khai ứng dụng Python. Nó giúp ta tránh được các sự cố do xung đột cấu hình hoặc thiếu thư viện.

Điểm nổi bật thứ hai là khả năng di động khi Image Docker có thể được dễ dàng di chuyển giữa các môi trường khác nhau, chẳng hạn như máy tính cá nhân, máy chủ đám mây hoặc thiết bị IoT. Điều này giúp ta dễ dàng triển khai ứng dụng Python của mình lên bất kỳ môi trường nào mà không cần phải lo lắng về việc cài đặt các phụ thuộc hoặc cấu hình lại ứng dụng.

Đồng thời là khả năng mở rộng với việc Docker giúp ta dễ dàng mở rộng ứng dụng Python của mình bằng cách tạo thêm container. Điều này rất hữu ích cho các ứng dụng cần xử lý nhiều dữ liệu hoặc lưu lượng truy cập cao.

Docker còn cung cấp các công cụ để dễ dàng quản lý các container, chẳng hạn như khởi động, dừng, khởi động lại và gỡ bỏ. Ta cũng có thể tự động hóa việc quản lý container bằng các công cụ như Docker Compose hoặc Kubernetes. Bên cạnh đó ứng dụng Python còn được bảo vệ khỏi môi trường hệ thống và các ứng dụng khác, tăng cường tính bảo mật nhưng việc chia sẻ ứng dụng Python vẫn rất dễ dàng với các thành viên trong nhóm và cộng đồng.

Tuy nhiên, cũng có một số hạn chế khi sử dụng Docker là phương pháp này có thể phức tạp hơn so với các phương pháp triển khai truyền thống. Container Docker có thể có overhead nhỏ so với các ứng dụng truyền thống. Và Docker có thể khiến người dùng phụ thuộc vào hệ sinh thái Docker.

Bất chấp những hạn chế này, Docker vẫn là một lựa chọn tuyệt vời cho việc triển khai ứng dụng Python. Nếu đang tìm kiếm một cách thức để triển khai ứng dụng Python của mình một cách nhất quán, di động, có thể mở rộng, hiệu quả và dễ dàng quản lý, thì Docker là lựa chọn phù hợp.

2.2.3 Phương pháp và tài nguyên sử dụng

2.2.3.1 Các mô hình học máy

a. Maxent Model: Logistics Regression

Mô hình MaxEnt (Maximum Entropy) là một thuật toán phân loại học máy hiệu quả được sử dụng rộng rãi trong xử lý ngôn ngữ tự nhiên (NLP), đặc biệt là trong các bài toán liên quan đến phân tích tình cảm và đánh giá sản phẩm.

Entropy là một khái niệm trong toán học và thống kê, được sử dụng để đo lường mức độ bất định hoặc rối loạn của một hệ thống. Trong mô hình MaxEnt, entropy được tính toán dựa trên phân phối xác suất của các lớp (label) trong tập dữ liệu.

Giả sử ta có tập dữ liệu X gồm N mẫu dữ liệu, được chia thành C lớp. Phân phối xác suất của lớp y cho dữ liệu x được ký hiệu là $p(y|x)$. Entropy của phân phối $p(y|x)$ được tính toán bằng công thức:

$$H(Y|X) = - \sum_y p(y|x) \log p(y|x)$$

Entropy càng cao, nghĩa là mức độ bất định của phân phối càng lớn.

MaxEnt hoạt động dựa trên nguyên tắc Entropy cực đại. Trong đó nguyên tắc Entropy cực đại cho rằng mô hình tốt nhất là mô hình tương thích nhất với dữ liệu huấn luyện, đồng thời ít giả định nhất về các dữ liệu chưa được quan sát. Nói cách khác, mô hình sẽ lựa chọn cách phân loại dữ liệu sao cho mức độ bất định (entropy) của phân phối xác suất là tối đa. Entropy được tính toán dựa trên phân phối xác suất của các lớp (label) trong tập dữ liệu.

Ưu điểm của mô hình:

- **Có khả năng học hỏi từ dữ liệu không có nhãn:** MaxEnt có thể học hỏi từ dữ liệu không có nhãn, điều này rất hữu ích trong các trường hợp dữ liệu huấn luyện có nhãn khan hiếm.
- **Có thể xử lý tốt dữ liệu nhiều và không đầy đủ:** MaxEnt có khả năng xử lý tốt dữ liệu nhiều và không đầy đủ, điều này thường xảy ra trong dữ liệu văn bản.
- **Dễ dàng thực hiện và giải thích:** MaxEnt là một mô hình tương đối đơn giản, dễ dàng thực hiện và giải thích.

Nhược điểm của mô hình:

- **Có thể bị ảnh hưởng bởi nhiều trong dữ liệu:** MaxEnt có thể bị ảnh hưởng bởi nhiều trong dữ liệu, dẫn đến kết quả phân loại không chính xác.
- **Có thể học thuộc dữ liệu huấn luyện:** MaxEnt có thể học thuộc dữ liệu huấn luyện, dẫn đến hiệu suất kém trên dữ liệu mới.

Mô hình Logistic Regression là một dạng cụ thể của Maxent Model. Trong hồi quy logistic, chúng ta áp dụng xạ đầu ra của mô hình vào một phạm vi giữa 0 và 1, thường được hiểu như xác suất. Điều này làm cho nó phù hợp cho các bài toán phân loại nhị phân, nơi chúng ta quan tâm đến việc ước tính xác suất thuộc một lớp cụ thể.

Trong bài toán phân loại với Maxent Model: Logistic Regression, chúng ta ước tính xác suất của việc một quan sát thuộc vào từng lớp, sau đó chọn lớp có xác suất cao nhất làm dự đoán. Thuật ngữ "logistic" xuất phát từ hàm sigmoid logistic được sử dụng để chuyển đổi kết quả của hồi quy tuyến tính thành một phạm vi giữa 0 và 1. Một ví dụ điển hình là phân loại Email, gồm có email công việc, email gia đình, email spam, ... Hay cụ thể hơn là ở bài toán này, phân loại đánh giá dựa trên bình luận của khách hàng về sản phẩm điện thoại di động trên sàn thương mại điện tử.

b. Machine Learning: Naive Bayes

Naive Bayes Classification (NBC) là một thuật toán phân loại dựa trên tính toán xác suất áp dụng định lý Bayes. Thuật toán này thuộc nhóm Supervised Learning (Học có giám sát). Có 2 mô hình thuật toán Naive Bayes thường sử dụng là: mô hình Bernoulli và mô hình Multinomial. Trong bài toán phân loại, Naive Bayes giả định tính độc lập giữa các đặc trưng, điều này có nghĩa là giá trị của mỗi đặc trưng không phụ thuộc vào giá trị của các đặc trưng khác khi đã biết nhãn lớp. Giả định này giúp đơn giản hóa tính toán và làm tăng tốc độ học của mô hình.

Trong bài toán này ta chỉ tìm hiểu mô hình Multinomial. Đặc trưng đầu vào ở đây chính là tần suất xuất hiện của từ trong văn bản đó. Tương tự như ý tưởng của nhóm là thực hiện tìm kiếm từ xuất hiện nhiều trong bình luận của khách hàng để phân loại ra bình luận đó có nội dung liên quan đến đánh giá bao nhiêu sao. Multinomial Naive Bayes được chọn vì nó phù hợp với bài toán phân loại nhiều lớp, thích hợp cho trường hợp đánh giá bình luận theo nhiều mức độ sao khác nhau. Mô hình này hoạt động dựa trên giả định rằng mỗi đặc trưng (từ trong văn bản) được mô tả bằng phân phối đa thức, và nó cực kỳ hiệu quả trong việc xử lý văn bản và dữ liệu đếm. Điều này làm cho Multinomial Naive Bayes là một lựa chọn tự nhiên cho bài toán của bạn.

2.2.3.2 Các phương pháp sử dụng

a. CountVectorizer

CountVectorizer là một lớp trong thư viện scikit-learn của Python được sử dụng để chuyển đổi văn bản thành dạng biểu diễn vectơ số. Nó được sử dụng phổ biến trong các ứng dụng xử lý ngôn ngữ tự nhiên (NLP) như phân loại văn bản, phân cụm văn bản và trích xuất đặc điểm.

CountVectorizer hoạt động bằng cách thực hiện các bước sau:

- Phân tách văn bản:** Chia văn bản thành các từ riêng lẻ (tokenization).
- Loại bỏ các từ dừng (stop words):** Loại bỏ các từ phổ biến không mang nhiều ý nghĩa như “lá”, “và”, v.v.
- Chuyển đổi từ thành số:** Gán một số duy nhất cho mỗi từ còn lại (indexing).
- Tạo ma trận vectơ:** Tạo một ma trận vectơ, với mỗi hàng đại diện cho một văn bản và mỗi cột đại diện cho một từ. Các giá trị trong ma trận là số lần xuất hiện của mỗi từ trong văn bản tương ứng.

CountVectorizer là một công cụ hữu ích trong xử lý ngôn ngữ tự nhiên, giúp chuyển đổi văn bản thành dạng vectơ số để sử dụng cho các ứng dụng như phân loại văn bản, phân cụm văn bản và trích xuất đặc điểm. Tuy nhiên, cần sử dụng nó một cách cẩn thận và kết hợp với các kỹ thuật khác để đạt hiệu quả tốt nhất.

b. TF-IDF Vectorizer

TF-IDF Vectorizer (viết tắt của Term Frequency-Inverse Document Frequency Vectorizer) là một lớp trong thư viện scikit-learn của Python được sử dụng để chuyển đổi văn bản thành dạng biểu diễn vectơ số, tương tự như CountVectorizer. Tuy nhiên, TF-IDF Vectorizer cải thiện hiệu quả của CountVectorizer bằng cách tính đến cả tần suất xuất hiện của từ (TF) và mức độ phổ biến của từ (IDF) trong tập văn bản.

TF-IDF Vectorizer hoạt động tương tự như CountVectorizer nhưng có thêm hai bước sau:

- Tính toán TF (Term Frequency):** Đo lường tần suất xuất hiện của mỗi từ trong văn bản tương ứng.
- Tính toán IDF (Inverse Document Frequency):** Đo lường mức độ phổ biến của mỗi từ trong tập văn bản. Một từ xuất hiện trong nhiều văn bản sẽ có IDF thấp, và ngược lại.

TF-IDF Vectorizer là một công cụ mạnh mẽ trong xử lý ngôn ngữ tự nhiên, giúp chuyển đổi văn bản thành dạng vector số có tính đến cả tần suất xuất hiện và mức độ phổ biến của từ. Nó giúp cải thiện hiệu quả của các mô hình học máy trong các ứng dụng như phân loại văn bản, phân cụm văn bản, trích xuất đặc điểm và hệ thống đề xuất. Tuy nhiên, cần sử dụng nó một cách cẩn thận và kết hợp với các kỹ thuật khác để đạt hiệu quả tốt nhất.

c. *Undersampling*

Undersampling là một phương pháp cân bằng dữ liệu bằng cách giảm số lượng mẫu trong lớp đa số. Phương pháp này giúp giảm thiểu ảnh hưởng của lớp đa số đến mô hình học máy, từ đó cải thiện hiệu quả phân loại cho lớp thiểu số. Undersampling thực hiện lấy mẫu ngẫu nhiên: Lấy mẫu ngẫu nhiên từ lớp đa số cho đến khi số lượng mẫu của lớp đa số bằng với số lượng mẫu của lớp thiểu số. Sau đó loại bỏ các mẫu dư thừa: Loại bỏ các mẫu còn lại của lớp đa số.

Phương pháp này đơn giản và dễ hiểu, nhanh chóng và hiệu quả về mặt tính toán, giảm thiểu nguy cơ overfitting. Tuy nhiên nó có thể loại bỏ thông tin quan trọng từ lớp đa số và có thể làm giảm độ chính xác của mô hình cho lớp đa số.

d. *Random Over Sampler*

Random Over Sampler (ROS) là một phương pháp cân bằng dữ liệu lấy mẫu ngẫu nhiên từ lớp đa số, dựa trên ý tưởng "oversampling thông minh". Phương pháp này giúp tăng cường dữ liệu cho lớp thiểu số một cách hiệu quả, đồng thời giảm nguy cơ overfitting.

ROS thực hiện các bước:

- Lấy mẫu ngẫu nhiên: Lấy mẫu ngẫu nhiên từ lớp đa số.
- Tính toán khoảng cách: Tính toán khoảng cách giữa mỗi mẫu được lấy mẫu ngẫu nhiên và các mẫu khác trong lớp đa số.
- Loại bỏ các mẫu gần: Loại bỏ các mẫu được lấy mẫu ngẫu nhiên có khoảng cách gần với các mẫu khác trong lớp đa số.
- Lặp lại: Lặp lại các bước trên cho đến khi số lượng mẫu của lớp đa số bằng với số lượng mẫu của lớp thiểu số.

Phương pháp này giúp tăng cường dữ liệu cho lớp thiểu số một cách hiệu quả, giảm nguy cơ overfitting, đơn giản và dễ hiểu. Nhưng cũng có thể loại bỏ thông tin quan trọng từ lớp đa số và có thể làm giảm độ chính xác của mô hình cho lớp đa số.

Nhìn chung ROS có một số ưu điểm quan trọng so với phương pháp oversampling thông thường như: ROS giúp giảm nguy cơ overfitting bằng cách loại bỏ các mẫu gần nhau trong lớp đa số. ROS có thể giúp cải thiện hiệu quả phân loại cho cả hai lớp, trong khi oversampling thông thường có thể làm giảm hiệu quả phân loại cho lớp đa số.

2.2.3.3 Các thư viện sử dụng

a. *Pandas*

Pandas là một thư viện mạnh mẽ cho việc thao tác và phân tích dữ liệu trong Python. Nó cung cấp cấu trúc dữ liệu linh hoạt, chủ yếu là DataFrame, giúp dễ dàng quản lý và phân tích dữ liệu. Pandas hỗ trợ các thao tác như lọc, tổng hợp và biến đổi dữ liệu, làm cho nó trở thành một công cụ không thể thiếu cho khoa học dữ liệu và học máy.

b. *String và Re*

String và Re (Regular Expressions) là các thư viện tiêu chuẩn của Python dùng để xử lý và thao tác chuỗi. Thư viện string cung cấp các hằng số và hàm hữu ích cho việc thao tác chuỗi, trong khi re cho phép làm việc với biểu thức chính quy, giúp tìm kiếm, so khớp và thay thế các mẫu chuỗi phức tạp.

c. Joblib

Joblib là một thư viện tối ưu hóa để lưu trữ và tải các đối tượng Python, đặc biệt là các mô hình học máy đã được huấn luyện. Joblib giúp tiết kiệm thời gian và tài nguyên bằng cách cung cấp cơ chế lưu trữ hiệu quả cho các đối tượng lớn.

d. Underthesea

Underthesea là một thư viện xử lý ngôn ngữ tự nhiên (NLP) cho tiếng Việt. Nó cung cấp các công cụ để phân tích và xử lý văn bản tiếng Việt như phân tích cú pháp, tách từ, gán nhãn từ loại, và nhiều chức năng khác, giúp cho việc xây dựng các ứng dụng NLP tiếng Việt trở nên dễ dàng hơn.

e. Scikit-learn

Scikit-learn là một trong những thư viện học máy phổ biến nhất trong Python, cung cấp các công cụ đơn giản và hiệu quả cho việc khai thác và phân tích dữ liệu. Các mô-đun được sử dụng trong đoạn mã bao gồm:

- CountVectorizer: Chuyển đổi văn bản thành ma trận các token (từ) được đếm.
- TfidfTransformer: Biến đổi một ma trận đếm từ thành ma trận TF-IDF, giúp xác định tầm quan trọng của từ trong văn bản.
- Train_test_split: Chia dữ liệu thành tập huấn luyện và tập kiểm tra, hỗ trợ quá trình đánh giá mô hình.

f. Openpyxl

Openpyxl là một thư viện mã nguồn mở được viết bằng Python, chuyên đọc, viết và thao tác với tệp Excel. Nó cung cấp nhiều chức năng hữu ích giúp đọc dữ liệu từ các bảng tính, ô tính. Giúp thao tác với dữ liệu và thư viện này hỗ trợ nhiều định dạng Excel phổ biến như .xlsx, .xslm, .xlsb. Ta có thể đọc và viết tệp Excel với bất kỳ định dạng nào trong số này.

g. Matplotlib

Matplotlib là một thư viện mã nguồn mở được viết bằng Python để tạo đồ thị và hình ảnh chất lượng cao hai chiều và ba chiều. Nó được sử dụng rộng rãi trong khoa học, kỹ thuật và các lĩnh vực khác để trực quan hóa dữ liệu.

h. Gradio

Gradio là một thư viện giúp xây dựng giao diện người dùng đơn giản cho các mô hình học máy và các ứng dụng dữ liệu. Nó cho phép tạo ra các giao diện web nhanh chóng và dễ dàng, hỗ trợ người dùng tương tác với các mô hình và kết quả phân tích mà không cần kỹ năng lập trình phức tạp.

Các thư viện Python được sử dụng cung cấp các chức năng mạnh mẽ và hiệu quả cho các tác vụ xử lý dữ liệu, thao tác chuỗi, biểu thức chính quy, lưu trữ dữ liệu và xử lý ngôn ngữ tiếng Việt. Nhờ có các thư viện này, việc phát triển các ứng dụng xử lý văn bản và học máy trở nên dễ dàng và hiệu quả hơn.

Chương 3. TRIỂN KHAI THỬ NGHIỆM HỆ THỐNG

3.1 ĐỀ XUẤT MÔ HÌNH TRIỂN KHAI

3.1.1 Xây dựng ứng dụng Python

3.1.1.1 Các bước thực hiện

Bước 1: Đánh giá tổng quan bộ dữ liệu đầu vào.

- Loại bỏ những cột không cần thiết.
- Tính số lượng của mỗi đánh giá.

Bước 2: Khám phá dữ liệu (Data Exploration)

Hiểu trước các đặc điểm của dữ liệu sẽ cho phép ta xây dựng mô hình phân lớp tốt hơn (nghĩa là đạt được độ chính xác cao hơn). Qua khám phá dữ liệu sơ bộ, ta thấy dữ liệu có đặc điểm mất cân bằng về phân phối các lớp đánh giá. Bên cạnh đó là sự xuất hiện của các từ thừa hoặc không mang nhiều ý nghĩa. Do đó, cần thực hiện các bước xử lý, làm sạch và cân bằng dữ liệu trong quá trình tiền xử lý. Sau khi xử lý, dữ liệu sẽ phù hợp để xây dựng các mô hình máy học hơn.

Bước 3: Tiền xử lý dữ liệu (Data Preprocessing)

- Tạo ra các hàm chức năng để làm sạch và xử lý dữ liệu văn bản, bao gồm: chuẩn hóa văn bản, xử lý lấy âm tiết, loại bỏ các common token, loại bỏ dấu câu và ký tự xuống dòng, chuẩn hóa các từ viết tắt, xử lý các từ dính nhau, xử lý các từ không có nghĩa ít hơn 3 chữ, loại bỏ stopword.
- Kiểm tra dữ liệu sau khi làm sạch.
- Tách từ.
- Rút trích đặc trưng.

Bước 4: Ứng dụng mô hình phân lớp

Xây dựng và đánh giá mô hình bằng các thuật toán Machine Learning.

Chia tập dữ liệu thành tập huấn luyện và tập kiểm tra sử dụng thư viện Scikit-learn. Sau đó, chúng ta sử dụng phương pháp phổ biến trong xử lý ngôn ngữ tự nhiên là TF-IDF để vector hóa văn bản. Mô hình máy học được xây dựng là Naive Bayes. Thực hiện Grid Search để tìm ra bộ tham số tốt, sau đó đánh giá hiệu suất của mô hình bằng cách tạo bảng tóm tắt các điểm số phân loại như Precision, Recall, và F1-score cho mỗi mô hình.

Xây dựng và đánh giá mô hình bằng các thuật toán Maximum Entropy.

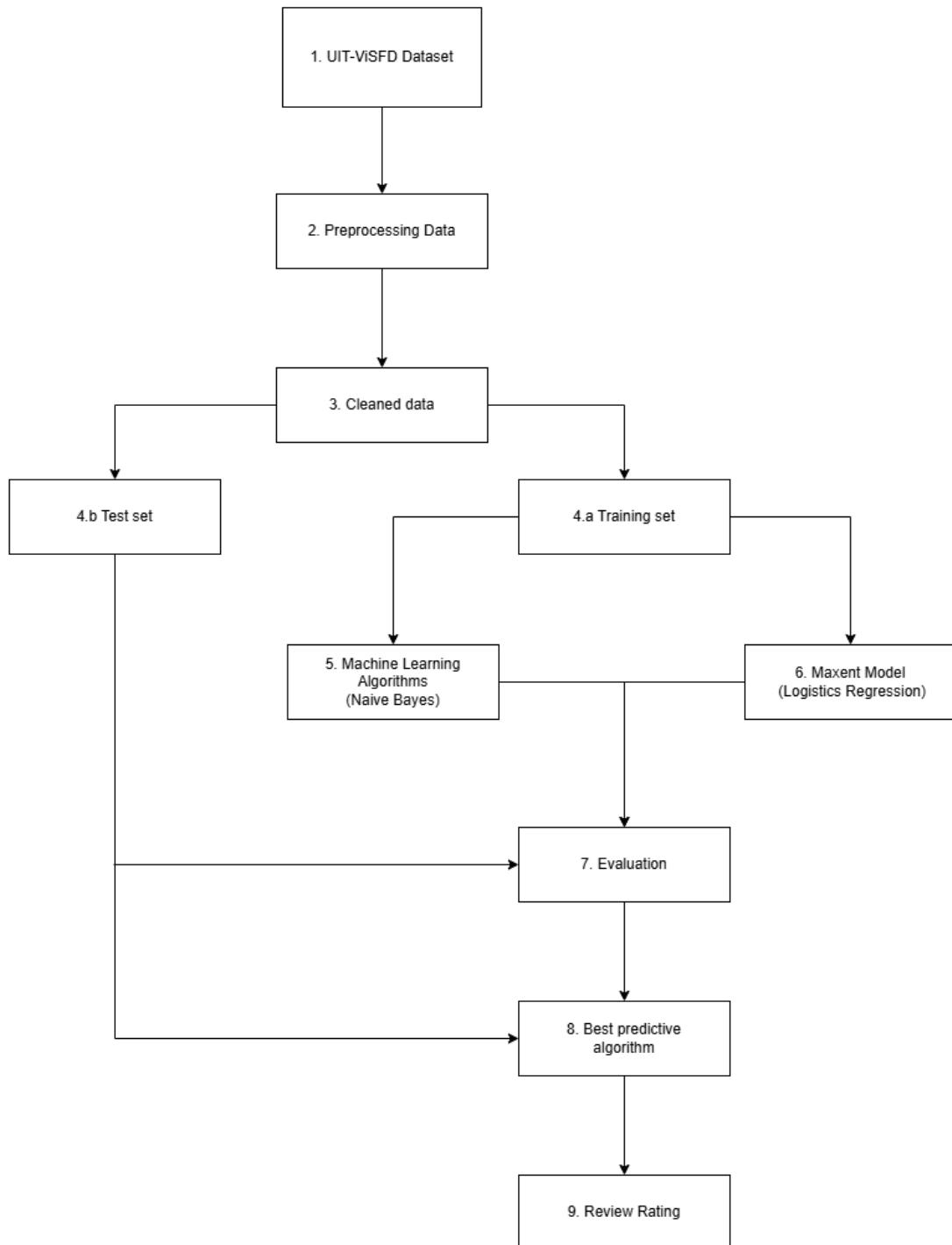
Tiếp theo, nhóm áp dụng thuật toán Maximum Entropy với mô hình Logistics Regression để phân loại lượt đánh giá sản phẩm. Quy trình xây dựng và đánh giá mô hình tương tự như trong phần trước, với việc sử dụng CountVectorizer và TF-IDF transformer để vector hóa văn bản và đánh giá mô hình bằng các độ đo hiệu suất trên tập kiểm tra.

Đây là quy trình tổng thể để xây dựng và đánh giá mô hình trên 2 thuật toán Machine Learning: Naive Bayes, Logistics Regression (Maximum Entropy). Thực hiện đánh giá 2 mô hình trên bằng các độ đo như Accuracy, Precision, Recall, F1 Score và

ma trận nhầm lẩn, chúng ta có thể lựa chọn mô hình tối ưu nhất giữa Logistic Regression, Naive Bayes. Sau đó xuất mô hình tốt nhất để sử dụng về sau.

3.1.1.2 Phương hướng triển khai thực nghiệm

Nhóm thực hiện triển khai phương hướng thực hiện theo lưu đồ như sau:



Hình 3.1-1 Phương hướng xây dựng ứng dụng Python

3.1.2 Triển khai ứng dụng Python trên Docker Image Official

3.1.2.1 Các bước thực hiện

Bước 1: Cài đặt Docker.

Cài đặt Docker trên máy tính, thực hiện các bước cài đặt phù hợp với hệ điều hành theo hướng dẫn trên trang chủ của Docker

Bước 2: Tự build image từ Dockerfile

- Tạo một thư mục cho dự án.
- Tạo file Dockerfile trong thư mục của dự án.
- Sao chép các tệp cần thiết đã xử lý vào thư mục của dự án.
- Build image từ Dockerfile.
- Chạy container từ image vừa build.

Bước 3: Tiến hành truy cập và chạy thử nghiệm

- Truy cập và chạy thử nghiệm trên local URL.
- Truy cập và chạy thử nghiệm trên public URL.

Bước 4: Tiến hành thử nghiệm khi tắt container và chạy lại container

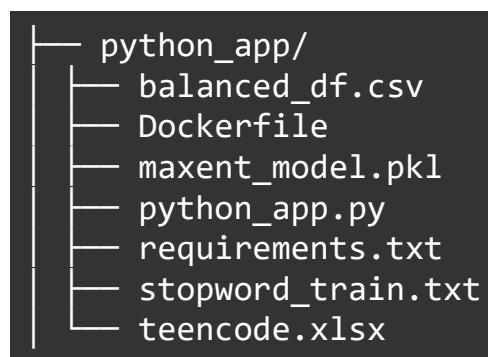
- Thủ nghiệm khi tắt container.
- Thủ nghiệm khi chạy lại container.

Bước 5: Đẩy Docker Image lên Docker Hub

- Đăng nhập vào Docker Hub.
- Kiểm tra ID Image cần đẩy lên.
- Đẩy Image lên Docker Hub

3.1.2.2 Cấu trúc của thư mục dự án

Các tệp và thư mục trong dự án được tổ chức như sau:



Trong đó:

- Dockerfile: Định nghĩa môi trường chạy ứng dụng Python trên Docker.
- python_app.py: Chứa mã nguồn chính của ứng dụng.
- requirements.txt: Liệt kê các thư viện Python cần thiết.
- balanced_df.csv: Dữ liệu cân bằng cho mô hình học máy.
- maxent_model.pkl: Mô hình Maxent đã huấn luyện.

- stopword_train.txt: Danh sách stopword sử dụng trong tiền xử lý văn bản.
- teencode.xlsx: Danh sách từ viết tắt teencode hỗ trợ biên dịch bình luận.

3.2 XÂY DỰNG ỨNG DỤNG PYTHON

3.2.1 Tổng quát bộ dữ liệu

Bộ dữ liệu về phản hồi đối với sản phẩm điện thoại thông minh của người Việt - UIT-ViSFD. Bộ dữ liệu gồm có 5 cột và 8898 dòng.

Kiểu dữ liệu trong dataset bao gồm 3 cột kiểu Categorical và 2 cột kiểu Numerical.

Bảng 3.2-1 Tổng quát bộ dữ liệu UIT-ViSFD

| Tên tính | thuộc | Kiểu dữ liệu | Mô tả |
|-----------|--------|--|-------|
| index | int64 | số thứ tự các bình luận | |
| comment | object | các bình luận của khách hàng và người tiêu dùng | |
| n_star | int64 | đánh giá của khách hàng và người tiêu dùng | |
| date_time | object | thời gian của các bình luận và đánh giá | |
| label | object | đánh giá tích cực/ tiêu cực/ trung tính theo các khía cạnh | |

Đối với n_star: 1 là cấp độ thấp nhất, thể hiện cho bình luận tiêu cực, 5 là cấp độ cao nhất, thể hiện bình luận tích cực.

| index | | comment | n_star | date_time | label |
|-------|---|---|--------|--------------|---|
| 0 | 0 | Mới mua máy này Tại thegioididong thốt nốt cảm... | 5 | 2 tuần trước | {CAMERA#Positive};{FEATURES#Positive};{BATTERY... |
| 1 | 1 | Pin kém còn lại miễn chê mua 8/3/2019 tinh trâ... | 5 | 14/09/2019 | {BATTERY#Negative};{GENERAL#Positive};{OTHERS}; |
| 2 | 2 | Sao lúc gọi điện thoại màn hình bị chấm nhỏ nh... | 3 | 17/08/2020 | {FEATURES#Negative}; |
| 3 | 3 | Mọi người cập nhật phần mềm lại, nó sẽ bớt tồ... | 3 | 29/02/2020 | {FEATURES#Negative};{BATTERY#Neutral};{GENERAL... |
| 4 | 4 | Mới mua Sài được 1 tháng thấy pin rất trâu, Sà... | 5 | 4/6/2020 | {BATTERY#Positive};{PERFORMANCE#Positive};{SER... |

Hình 3.2-1 Bộ dữ liệu UIT-ViSFD

3.2.2 Tiền xử lý dữ liệu

3.2.2.1 Loại bỏ các cột không cần thiết

Nhóm chỉ thực hiện phân loại đánh giá dựa trên bình luận của khách hàng, vì vậy quyết định drop đi các cột còn lại.

```
df.drop(['index', 'date_time', 'label'], axis=1, inplace=True)
```

Tổng quát bộ dữ liệu sau khi đã xóa các cột, có 8898 dòng dữ liệu và còn lại 2 cột comment và n_star

Bảng 3.2-2 Tổng quát bộ dữ liệu UIT-ViSFD sau khi xóa cột

| Tên thuộc tính | Kiểu dữ liệu | Mô tả | Ghi chú |
|----------------|--------------|---|------------------------------|
| comment | object | các bình luận của khách hàng và người tiêu dùng | |
| n_star | int64 | đánh giá của khách hàng và người tiêu dùng | Đánh giá theo thang điểm 1-5 |

3.2.2.2 Chuẩn hóa văn bản tiếng Việt

Chuẩn hóa văn bản tiếng Việt, để chuẩn hóa kiểu gõ dấu theo kiểu cũ, chuẩn hóa bảng mã Unicode để thống nhất với nhau.

3.2.2.3 Xử lý lấy âm tiết

Tạo hàm chức năng có tên là **handle_repeated_syllables** để xóa các từ láy âm tiết, Sử dụng regex để tìm các từ có âm tiết lặp lại (ví dụ: quááááá -> quá), chỉ giữ lại một phần và thêm vào từ gốc.

```
def handle_repeated_syllables(text):
    # Sử dụng regex để tìm các từ có âm tiết lặp lại (ví dụ: quááááá)
    repeated_syllables_pattern = re.compile(r'(\w+?)\1+', re.UNICODE)
    # Hàm xử lý việc loại bỏ âm tiết lặp lại
    def handle_repetition(match):
        word = match.group(1)
        # Giữ lại chỉ một phần lặp lại và thêm vào từ gốc
        return word
```

3.2.2.4 Loại bỏ các common token

Tạo hàm **replace_common_token** để loại bỏ các common token như PHONE, MENTION, NUMBER, DATETIME, ... Đối với bộ dữ liệu này, đây là những thông tin không cần thiết nên bỏ đi bằng cách sử dụng regex.

```
def replace_common_token(txt):
    txt = re.sub(EMAIL, ' ', txt)
    txt = re.sub(URL, ' ', txt)
    txt = re.sub(MENTION, ' ', txt)
    txt = re.sub(DATETIME, ' ', txt)
    txt = re.sub(NUMBER, ' ', txt)
    txt = re.sub(PRICE, ' ', txt)
    return txt
```

3.2.2.5 Loại bỏ dấu câu và ký tự xuống dòng

Tạo hàm **remove_unnecessary_characters** bỏ dấu câu và các ký tự xuống dòng bằng cách sử dụng regex và string.punctuation.

```

def remove_unnecessary_characters(text):
    RE_CLEAR = re.compile("[\n\r]+")# Thay thế các chuỗi xuống
dòng (\n hoặc \r) bằng một ký tự trắng
    text = re.sub(RE_CLEAR, ' ', text)
    # Sử dụng string.punctuation để lấy tất cả các ký tự dấu
câu
    translator = str.maketrans(' ', ' ', string.punctuation)
    # Loại bỏ dấu câu từ văn bản sử dụng bảng dịch
(translator)
    text = text.translate(translator)
    return text

```

3.2.2.6 Chuẩn hóa các từ viết tắt cơ bản (*r -> rồi, đc -> được*)

Chuẩn hóa các từ viết tắt cơ bản bằng hàm ***normalize_acronyms***. Tải bộ từ viết tắt [teencode tham khảo](#) (bao gồm 1 cột từ viết tắt và 1 cột từ gốc) và sau đó thay những từ viết tắt của dữ liệu có trong bộ từ viết tắt bằng từ gốc trong bộ từ viết tắt.

```

def normalize_acronyms(text,
teencode_file=folder+'/Data/teencode.xlsx'):
    # Đọc dữ liệu từ tệp Excel teencode.xlsx
    teencode_df = pd.read_excel(teencode_file, header=None,
names=['teencode', 'replace'])
    words = []
    for word in text.strip().split():
        word = word.strip(string.punctuation)
        # Tìm kiếm trong teencode_df và thay thế
        replacement =
teencode_df.loc[teencode_df['teencode'].str.lower() == word,
'replace'].values
        if len(replacement) > 0:
            words.append(replacement[0])
        else:
            words.append(word)
    return ' '.join(words)

```

3.2.2.7 Xử lý các từ viết dính nhau bằng cách tách ra (*hoinhanh -> hơi nhanh*)

Lấy các từ không nằm trong bộ từ đơn tiếng Việt có độ dài hơn 6 (vì từ đơn tiếng Việt dài nhất là 6 chữ). Tiến hành chia những từ này ra làm 2 phần, xét nếu 2 phần đều thuộc bộ từ đơn tiếng Việt thì tách ra bằng khoảng trắng, sau đó thu được bộ từ bao gồm 2 cột, 1 cột là các từ bị dính nhau và cột thứ 2 là các từ dính nhau được tách ra.

```

def stuck_words(text):
    for i in range(1,len(text)):
        word1 = text[:i]
        word2 = text[i:]
        if word1 in word_dict and word2 in word_dict:
            text_after = word1 + ' ' + word2

```

```

        return text_after
    break
# Gán các giá trị có thể tách ra vào cột mới stuck_word
not_vietnamese_than_6['stuck_word'] = float('nan')
for i in not_vietnamese_than_6['word']:
    not_vietnamese_than_6.loc[not_vietnamese_than_6['word'] == i, 'stuck_word'] = stuck_words(i)

```

Tạo hàm ***normalize_stuck*** bằng cách dựa vào bộ từ này để tách các từ bị dính nhau trong dữ liệu, hàm sử dụng tương tự với chuẩn hóa những viết tắt.

```

def normalize_stuck(text, df_stuck_words=df_stuck_words):
    words = []
    for word in text.strip().split():
        word = word.strip(string.punctuation)
        # Tìm kiếm trong teencode_df và thay thế
        replacement =
df_stuck_words.loc[df_stuck_words['word'].str.lower() == word,
'stuck_word'].values
        if len(replacement) > 0:
            words.append(replacement[0])
        else:
            words.append(word)

    return ' '.join(words)

```

3.2.2.8 Loại bỏ các stopword

Tạo hàm ***remove_stopwords*** để loại bỏ các stopword bằng cách tạo hàm ***get_stopwords*** để xác định những từ có idf thấp hơn một ngưỡng cụ thể trong bộ dữ liệu, và lấy danh sách này làm stopwords trong bộ dữ liệu. Tải về một bộ stopword tiếng Việt khác được lấy từ một số lượng lớn văn bản và so khớp, chỉ giữ lại những stopword đều thuộc 2 bộ stopword đã có sau đó bổ sung thêm tập not vietnamese less 3 ở trên vào. Tiến hành loại bỏ những từ có trong bộ stopword này. Cuối cùng xuất ra file ***stopword_train.txt***.

```

def get_stopwords(documents, threshold=3):
    """
    :param documents: list of documents
    :param threshold:
    :return: list of words has idf <= threshold
    """
    tfidf = TfidfVectorizer(min_df=100)
    tfidf_matrix = tfidf.fit_transform(documents)
    features = tfidf.get_feature_names_out()
    stopwords = []
    print(min(tfidf.idf_), max(tfidf.idf_), len(features))
    for index, feature in enumerate(features):
        if tfidf.idf_[index] <= threshold:

```

```

        stopwords.append(feature)
    return stopwords

```

```

def remove_stopwords(line):
    words = []
    for word in line.strip().split():
        if word not in stopword:
            words.append(word)
    return ' '.join(words)

```

3.2.2.9 Tạo hàm `text_preprocess` để tổng hợp lại các hàm chức năng ở trên và áp dụng cho dữ liệu input

```

def text_preprocess(text):
    # 1. Chuẩn hóa văn bản tiếng việt
    text = text_normalize(text)
    # 2. Xử lý lấy âm tiết
    text = handle_repeated_syllables(text)
    # 3. Loại bỏ các common token
    text = replace_common_token(text)
    # 4. Xóa bỏ dấu câu
    text = remove_unnecessary_characters(text)
    # 5. Đưa về lower
    text = text.lower()
    # 7. Chuẩn hóa các từ viết tắt cơ bản
    text = normalize_acronyms(text)
    # 8. Tách các từ viết dính nhau
    text = normalize_stuck(text)
    # 9. Loại bỏ các stopword tiếng Việt
    text = remove_stopwords(text)
    # 10. Loại bỏ các khoảng trắng liên tiếp
    RE_CLEAR = re.compile("\s+") # Các khoảng trắng liên tiếp
    text = re.sub(RE_CLEAR, ' ', text)
    return text

```

Lưu các dữ liệu đã được apply hàm `text preprocess2` vào 1 cột mới ‘comment clean2’ trong dataframe.

```

df["comment_clean2"] =
df['comment_clean'].apply(text_preprocess)

```

Sau đó tách từ trong câu bằng lệnh `word_tokenize` từ thư viện underthesea.

```

def word_segmentation(text):
    text = underthesea.word_tokenize(text, format="text")
    return text

```

Lưu các dữ liệu đã được apply hàm `word_segmentation` vào 1 cột mới ‘comment_token’ trong dataframe.

```
df['comment_token'] =  
df['comment_clean2'].apply(word_segmentation)
```

3.2.3 Huấn luyện mô hình

3.2.3.1 Xây dựng và đánh giá mô hình bằng các thuật toán máy học Naive Bayes

- **Chia tập dữ liệu**

Sử dụng thư viện Scikit-learn (sklearn) để chia tập dữ liệu thành tập huấn luyện và tập kiểm tra. Cụ thể, đầu vào của hàm `train_test_split` là 2 Series X và y. Với X chứa văn bản sau khi đã được xử lý và làm sạch, được lưu trữ trong cột '`comment_token`' của dataframe df. Và y là dữ liệu về nhãn của các văn bản, được lưu trữ trong cột '`n_star`' của dataframe df.

```
X, y = df['comment_token'], df['n_star']
```

Sử dụng hàm `train_test_split()` trong sklearn để tách dữ liệu thành tập huấn luyện và tập kiểm tra. Mặc định, hàm `train_test_split` sẽ chia dữ liệu đầu vào thành hai phần với tỷ lệ 80% cho tập huấn luyện và 20% cho tập kiểm tra. Ta sử dụng tham số `random_state` thiết lập giá trị ngẫu nhiên được sử dụng để xáo trộn dữ liệu trước khi chia. Bằng cách đặt giá trị này thành một số cụ thể (trong trường hợp này là 42), chúng ta có thể đảm bảo rằng cùng một chia ngẫu nhiên được tạo ra mỗi lần chạy mã. Điều này hữu ích cho tính nhất quán trong quá trình tái tạo.

Kết quả trả về của hàm `train_test_split` là 4 mảng `x_train`, `x_test`, `y_train`, `y_test` chứa dữ liệu đã được chia thành tập huấn luyện và tập kiểm tra tương ứng.

```
X_train, X_test, y_train, y_test = train_test_split(X, y,  
test_size=0.2, random_state=42)
```

- **Vector hóa văn bản**

Ta dùng đến phương pháp TF-IDF được sử dụng trong xử lý ngôn ngữ tự nhiên để vector hóa văn bản, sử dụng 2 hàm CountVectorizer và TF-IDF transformer.

- **CountVectorizer:** sử dụng CountVectorizer để chuyển đổi văn bản thành ma trận đếm các từ, bỏ qua các từ trong danh sách `stop_words` (từ không có ý nghĩa như "và", "với", ...), và chỉ tính toán các từ xuất hiện ít nhất `min_df` lần trong toàn bộ tập dữ liệu.
- **TF-IDF transformer:** được sử dụng để chuyển đổi ma trận đếm từ (CountVectorizer tạo ra) thành ma trận TF-IDF.

```
count_vectorizer = CountVectorizer(ngram_range=(1, 5),  
stop_words=stopword, max_df=0.5, min_df=5)  
tfidf_vectorizer = TfidfTransformer(use_idf=False,  
sublinear_tf = True, norm='l2', smooth_idf=True)  
  
X_train = count_vectorizer.fit_transform(X_train)  
X_train = tfidf_vectorizer.fit_transform(X_train)  
  
X_test = count_vectorizer.transform(X_test)  
X_test = tfidf_vectorizer.transform(X_test)
```

- **Xây dựng mô hình bằng thuật toán máy học Naive Bayes**

Sử dụng MultinomialNB để xây dựng một mô hình phân loại Naive Bayes dựa trên các vector đặc trưng và thực hiện Grid Search để tìm ra bộ tham số tốt nhất cho mô hình.

```
# Khởi tạo mô hình MultinomialNB
naive_bayes_classifier = MultinomialNB()

# Thiết lập các tham số cần tìm kiếm
parameters = {
    'force_alpha' : (True, False),
    'fit_prior': (True, False),
    'alpha': (1, 0.1, 0.01)
}

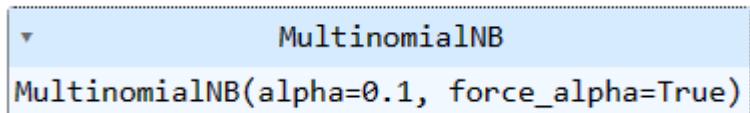
# Khởi tạo GridSearchCV với mô hình, tham số và số lượng fold
# trong cross-validation (cv)
grid_search_nb =
GridSearchCV(estimator=naive_bayes_classifier,
param_grid=parameters, cv=5, scoring='accuracy')

# Huấn luyện GridSearchCV trên dữ liệu
grid_search_nb.fit(X_train, y_train)

# In ra các tham số tốt nhất
print("Best Parameters:", grid_search_nb.best_params_)

Best Parameters: {'alpha': 0.1, 'fit_prior': True,
'force_alpha': True}
```

Sử dụng bộ tham số đó đưa vào mô hình:



Hình 3.2-2 Tham số mô hình Naive Bayes

- **Đánh giá mô hình MultinomialNB:**

Đánh giá hiệu suất của mô hình bằng các chỉ số đánh giá, sử dụng hàm *classification_report* báo cáo được in ra để cung cấp cái nhìn tổng quan về khả năng dự đoán của mô hình Naive Bayes trên tập kiểm tra. Các độ đo trong báo cáo giúp định lượng khả năng của mô hình trong việc phân loại các lớp khác nhau và đánh giá khả năng tổng quát của mô hình.

```
# Dự đoán trên tập test
y_pred_nb = nb_model.predict(X_test)
# Đánh giá mô hình bằng các chỉ số
report1 = metrics.classification_report(y_test, y_pred_nb,
```

| | | | | |
|-----------------------------|------------------------|---------------------|-----------------------|----------------------|
| <code>digits=3)</code> | | | | |
| <code>print(report1)</code> | | | | |
| | <code>precision</code> | <code>recall</code> | <code>f1-score</code> | <code>support</code> |
| 1 | 0.557 | 0.731 | 0.633 | 279 |
| 2 | 0.364 | 0.034 | 0.063 | 117 |
| 3 | 0.355 | 0.270 | 0.307 | 222 |
| 4 | 0.306 | 0.182 | 0.228 | 286 |
| 5 | 0.746 | 0.906 | 0.819 | 876 |
| <code>accuracy</code> | | | 0.626 | 1780 |
| <code>macro avg</code> | 0.466 | 0.425 | 0.410 | 1780 |
| <code>weighted avg</code> | 0.572 | 0.626 | 0.581 | 1780 |

Dựa trên báo cáo, có thể kết luận rằng mô hình có độ chính xác (accuracy) là khoảng 62.6%. Mô hình có hiệu suất khác biệt đáng kể giữa các lớp, được thể hiện qua các chỉ số precision, recall, và F1-score, kết quả cho mỗi lớp là không đồng đều. Tổng quát, mô hình có hiệu suất tốt đối với lớp 5, nhưng có hiệu suất thấp đối với các lớp khác, đặc biệt là lớp 2. Nguyên nhân là do lớp 5 chiếm nhiều phần trăm trong bộ dữ liệu, còn lớp 2 chỉ chiếm 7.3% nên hiệu suất của mô hình đối với lớp này thấp. bộ dữ liệu. Vì vậy cần xem xét xử lý mất cân bằng giữa các lớp trong dữ liệu.

Thực hiện *cross validation* để đánh giá biến động

```
#CROSS VALIDATION
cross_score = cross_val_score(nb_model, X_train,y_train,
cv=10)
cross_score

array([0.5744382 , 0.57022472, 0.55758427, 0.56460674,
0.56601124,
       0.56320225, 0.55617978, 0.56320225, 0.5625879 ,
0.58931083])
```

Các giá trị độ chính xác nằm trong khoảng từ khoảng 0.556 đến 0.589. Sự biến động này không quá lớn, có thể coi là đồng đều. Kiểm tra biến động giữa các fold cho thấy hiệu quả của mô hình trên nhiều tập dữ liệu khá giống nhau. Điều này có thể minh chứng mô hình không bị overfitting.

3.2.3.2 Xây dựng và đánh giá mô hình bằng các thuật toán Maxent

- **Chia tập dữ liệu**

Sử dụng cùng tập dữ liệu X_train, y_train, X_test, y_test với mô hình máy học Naive Bayes. Bộ dữ liệu đưa vào đã được làm sạch, tách từ và trích xuất đặc trưng.

- **Xây dựng mô hình bằng thuật toán máy học Logistic Regression**

Logistic regression cũng là một mô hình dựa trên nguyên lý Maximum entropy. Sử dụng Logistic regression để huấn luyện một mô hình phân loại đa lớp từ các vectơ đặc trưng và thực hiện Grid Search để tìm ra bộ tham số tốt nhất cho mô hình.

```

# Tạo mô hình Logistic Regression (Softmax Regression)
model = LogisticRegression()

# Định nghĩa các tham số cần tối ưu
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100],
              'penalty': ['l2'],
              'multi_class': ['multinomial'], # xác định rằng
đang thực hiện phân loại đa lớp (softmax regression).
              'solver' : ['lbfgs','sag','saga','newton-cg']
            }

# Tạo đối tượng GridSearchCV
grid_search = GridSearchCV(estimator=model,
param_grid=param_grid, cv=5, scoring='accuracy')

# Thực hiện grid search trên tập huấn luyện
grid_search.fit(X_train, y_train)

# Hiển thị kết quả tối ưu
print("Best parameters: ", grid_search.best_params_)

Best parameters: {'C': 1, 'multi_class': 'multinomial',
'penalty': 'l2', 'solver': 'lbfgs'}

```

Sử dụng bộ tham số đó đưa vào mô hình:

```

LogisticRegression
LogisticRegression(C=1, multi_class='multinomial')

```

Hình 3.2-3 Tham số mô hình Logistic Regression

- Đánh giá mô hình Logistic Regression:**

Đánh giá hiệu suất của mô hình bằng các chỉ số đánh giá, sử dụng hàm *classification_report* báo cáo được in ra để cung cấp cái nhìn tổng quan về khả năng dự đoán của mô hình Logistic Regression trên tập kiểm tra.

```

# Dự đoán trên tập kiểm tra
y_pred_lr = maxent_model.predict(X_test)
# Đánh giá mô hình bằng các chỉ số
report1 = metrics.classification_report(y_test, y_pred_lr,
digits=3)
print(report1)

precision      recall    f1-score   support

1            0.572     0.699     0.629      279
2            0.312     0.043     0.075      117
3            0.342     0.248     0.287      222
4            0.351     0.206     0.260      286

```

| | | | | |
|--------------|-------|-------|-------|------|
| 5 | 0.734 | 0.917 | 0.815 | 876 |
| accuracy | | | 0.628 | 1780 |
| macro avg | 0.462 | 0.422 | 0.413 | 1780 |
| weighted avg | 0.570 | 0.628 | 0.582 | 1780 |

Dựa trên báo cáo, có thể kết luận rằng mô hình có độ chính xác (accuracy) là khoảng 62.6%. Tổng thể, mô hình có vẻ hoạt động tốt đối với các lớp có đánh giá cao hơn (lớp 1 và lớp 5), nhưng hiệu suất giảm đáng kể đối với các lớp khác, đặc biệt là lớp 2 và lớp 4. Điều này có thể yêu cầu xem xét và điều chỉnh thêm mô hình để cải thiện hiệu suất trên các lớp khó dự đoán hơn.

Thực hiện *cross validation* để đánh giá biến động

```
#CROSS VALIDATION
cross_score = cross_val_score(maxent_model, X_train,y_train,
cv=10)
cross_score

array([0.60252809, 0.61376404, 0.58707865, 0.59691011,
0.5997191 ,
       0.59269663, 0.58426966, 0.6011236 , 0.59634318,
0.604782 ])
```

Các giá trị độ chính xác nằm trong khoảng từ khoảng 0.584 đến 0.613. Các giá trị này có vẻ ổn định, và có vẻ mô hình đang giữ được sự ổn định trên các tập dữ liệu khác nhau. Điều này là một dấu hiệu tích cực về tính tổng quát của mô hình.

3.2.4 Đánh giá mô hình

Với bộ dữ liệu trên, các kết quả đánh giá đều không quá tốt, đặc điểm chung của các kết quả trên là luôn hoạt động tốt đối với một vài lớp và không hiệu quả đối với các lớp còn lại. Nguyên nhân là do dữ liệu bị mất cân bằng, vì thế xử lý mất cân bằng dữ liệu bằng cách giảm kích thước mẫu hoặc tăng kích thước mẫu.

3.2.4.1 Giảm kích thước mẫu

Chia dữ liệu thành các lớp, sau đó xác định lớp có số lượng mẫu ít nhất và giảm mẫu các lớp khác để đảm bảo không vượt quá số lượng mẫu ít nhất. Kết hợp lại thành DataFrame mới cân bằng. Thu được bảng kết quả như sau với 3240 dòng, trong đó mỗi đánh giá là 648 dòng:

| index | Unnamed: 0 | comment_n_star | comment_clean | comment_clean2 | comment_length | comment_token |
|-------|------------|---|---|---|----------------|---|
| 0 | 7043 | Có ai hay tự nhiên bị mất wf như mình ko. Phải... | có ai hay tự nhiên bị mất wf như mình không ph... | có ai hay tự nhiên bị mất wf như mình không ph... | 148 | có ai hay_tự_nhiên bị mất wf như mình không ph... |
| 1 | 3698 | Thứ nhì, độ sáng 35 % không auto, xem Facebook... | thứ nhì độ sáng không auto xem facebook và nhâ... | thứ nhì độ sáng không auto xem facebook và nhâ... | 796 | thứ nhì độ sáng không auto xem facebook và nhâ... |
| 2 | 305 | Sài 1 thằng là thấy máy xuống cấp trầm trọng... | xài thang là thấy máy xuống cấp trầm trọng lỗi... | xài thang là thấy máy xuống cấp trầm trọng lỗi... | 280 | xài thang là thấy máy xuống_cấp trầm trọng lỗi... |
| 3 | 2125 | Mình có mua đt này chưa dc một thời mà chụp à... | mình có mua điện thoại này chưa dc một thời... | mình có mua điện thoại này chưa dc một tháng... | 111 | mình có mua điện_thoại này chưa dc một tháng... |
| 4 | 1253 | Mua 11/07/2020 đến giờ dùng nhanh hết pin và... | mua đến giờ dùng nhanh hết pin và vào mục gi... | mua đến giờ dùng nhanh hết pin và vào mục gi... | 76 | mua đến giờ dùng nhanh hết pin và vào mục gi... |
| ... | ... | ... | ... | ... | ... | ... |
| 3235 | 7374 | Quá tuyệt vời tôi mua được 3 ngày dùng ok lắm p... | quá tuyệt vời tôi mua được ngày dùng ok lắm p... | quá tuyệt vời tôi mua được ngày dùng ok lắm p... | 208 | quá tuyệt_với tôi mua được ngày dùng ok lắm p... |
| 3236 | 3689 | Pin khá tốt, dùng cả ngày tôi về vẫn còn gần 3... | pin khá tốt dùng cả ngày tôi về vẫn còn gần ch... | pin khá tốt dùng cả ngày tôi về vẫn còn gần ch... | 372 | pin khá tốt dùng cả ngày tôi về vẫn còn gần ch... |
| 3237 | 5663 | Máy ok về mọi thứ inCamera thì ko nét lắm inHe... | máy ok về mọi thứ camera thì không nét lắm anh... | máy ok về mọi thứ camera thì không nét lắm anh... | 122 | máy ok về mọi thứ camera thi không nét lắm anh... |
| 3238 | 6884 | Máy rất tốt a chơi game và làm việc đều được h... | máy rất tốt a chơi game và làm việc đều được h... | máy rất tốt a chơi game và làm việc đều được h... | 130 | máy rất tốt a chơi game và làm_viec đều được h... |
| 3239 | 6150 | hiệu năng ngon .hoi ấm .cảm giác cầm chơi lâu th... | hiệu năng ngon hơi ấm cảm giác cầm chơi lâu th... | hiệu năng ngon hơi ấm cảm giác cầm chơi lâu th... | 168 | hiệu_năng ngon_hơi ấm_cảm giác cầm chơi lâu th... |

3240 rows x 8 columns

Hình 3.2.4 Bộ dữ liệu UIT-ViSFD khi giảm mẫu

Dánh giá hiệu suất của mô hình bằng các chỉ số đánh giá, sử dụng hàm *classification_report* báo cáo được in ra để cung cấp cái nhìn tổng quan về khả năng dự đoán của 2 mô hình Logistic Regression và Naive Bayes trên tập kiểm tra, thu được kết quả như sau:

| Classification Report for Logistic Regression model: | | | | |
|--|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 1 | 0.584 | 0.510 | 0.544 | 157 |
| 2 | 0.311 | 0.288 | 0.299 | 132 |
| 3 | 0.264 | 0.320 | 0.289 | 122 |
| 4 | 0.277 | 0.267 | 0.272 | 116 |
| 5 | 0.550 | 0.587 | 0.568 | 121 |
| accuracy | | | 0.400 | 648 |
| macro avg | 0.397 | 0.394 | 0.394 | 648 |
| weighted avg | 0.407 | 0.400 | 0.402 | 648 |
| Classification Report for Naive Bayes model: | | | | |
| | precision | recall | f1-score | support |
| 1 | 0.540 | 0.478 | 0.507 | 157 |
| 2 | 0.311 | 0.318 | 0.315 | 132 |
| 3 | 0.291 | 0.320 | 0.305 | 122 |
| 4 | 0.349 | 0.319 | 0.333 | 116 |
| 5 | 0.604 | 0.669 | 0.635 | 121 |
| accuracy | | | 0.423 | 648 |
| macro avg | 0.419 | 0.421 | 0.419 | 648 |
| weighted avg | 0.424 | 0.423 | 0.423 | 648 |

Cả hai mô hình có khả năng dự đoán khác nhau đối với các lớp. Điều này có thể phản ánh sự hiệu quả của mỗi mô hình đối với từng lớp cụ thể. Naive Bayes có vẻ cho kết quả tốt hơn với một số chỉ số đánh giá (như precision và F1-score) so với Logistic Regression trên tập kiểm thử. Cả hai mô hình đều đạt hiệu quả cao hơn ở lớp 1 và 5, đạt hiệu quả thấp hơn ở 3 lớp còn lại. Kết quả đạt được của 2 mô hình trên bộ dữ liệu giảm mẫu tệ hơn trước khi giảm mẫu, nên đây không phải một phương pháp hiệu quả để cân bằng dữ liệu.

3.2.4.2 Tăng kích thước mẫu

Sử dụng RandomOverSampler để tăng kích thước mẫu. Thu được bảng kết quả như sau với 20925 dòng, trong đó mỗi đánh giá là 4185 dòng:

| index | Unnamed: 0 | comment | n_star | comment_clean | comment_clean2 | comment_length | comment_token |
|-------|------------|---|--------|--|--|----------------|--|
| 0 | 7043 | Có ai hay tự nhiên bị mất v汜 như mình ko Phải... | 1 | có ai hay tự nhiên bị mất v汜 như mình không ph... | có ai hay tự nhiên bị mất v汜 như mình không ph... | 148 | có ai hay tự_nhiên bị mất v汜 như mình không ph... |
| 1 | 3698 | Thứ nhất: độ sáng 35 % không auto_xem Faceboo... | 1 | thứ nhất độ sáng không auto xem facebook và nhâ... | thứ nhất độ sáng không auto xem facebook và nhâ... | 796 | thứ nhất độ sáng không auto_xem facebook và nhâ... |
| 2 | 305 | Sài 1 tháng là thấy máy xuống cấp trầm trọng... | 1 | xài tháng là thấy máy xuống cấp trầm trọng lỗi... | xài tháng là thấy máy xuống cấp trầm trọng lỗi... | 280 | xài tháng là thấy máy xuống_cấp trầm trọng lỗi... |
| 3 | 2125 | Mình có mua đt này chưa dc một tháng mà chụp ă... | 1 | mình có mua điện thoại này chưa được một tháng... | mình có mua điện thoại này chưa được một tháng... | 111 | mình có mua điện_thoại này chưa được một tháng... |
| 4 | 1253 | Mua 11/07/2020 đến giờ dùng nhanh hết pin và v... | 1 | mua đến giờ dùng nhanh hết pin và vào mục gi... | mua đến giờ dùng nhanh hết pin và vào mục gi... | 76 | mua đến giờ dùng nhanh hết pin và vào mục gi... |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 3235 | 7374 | Quá tuyệt vời tôi mua được 3 ngày dùng ok lắm ... | 5 | quá tuyệt vời tôi mua được ngày dùng ok lắm pi... | quá tuyệt vời tôi mua được ngày dùng ok lắm pi... | 208 | quá_tuyệt_với tôi mua được ngày dùng ok_lần pi... |
| 3236 | 3689 | Pin khá tốt, dùng cả ngày tôi về vẫn còn gần 3... | 5 | pin khá tốt dùng cả ngày tôi về vẫn còn gần ch... | pin khá tốt dùng cả ngày tôi về vẫn còn gần ch... | 372 | pin_khá_tốt dùng cả ngày tôi về vẫn còn gần ch... |
| 3237 | 5663 | Máy ok về mọi thứ inCamera thì ko nét lắm inAe... | 5 | máy ok về mọi thứ camera thi không nét lắm anh... | máy ok về mọi thứ camera thi không nét lắm anh... | 122 | máy_ok về mọi thứ camera thi không_nét lắm anh... |
| 3238 | 6884 | Máy rất tốt a chơi game và làm việc đều được h... | 5 | máy rất tốt a chơi game và làm việc đều được h... | máy rất tốt a chơi game và làm việc đều được h... | 130 | máy_rất_tốt a chơi game và làm việc đều được h... |
| 3239 | 6150 | hiệu năng ngon _hoi_ấm_cảm giác cầm chơi lâu ... | 5 | hiệu năng ngon hơi ấm cảm giác cầm chơi lâu th... | hiệu năng ngon hơi ấm cảm giác cầm chơi lâu th... | 158 | hiệu_năng_ngon_hơi_ấm_cảm giác cầm chơi lâu th... |

Hình 3.2-5 Bộ dữ liệu UIT-ViSFD khi tăng mẫu

Đánh giá hiệu suất của mô hình bằng các chỉ số đánh giá, sử dụng hàm *classification_report* báo cáo được in ra để cung cấp cái nhìn tổng quan về khả năng dự đoán của 2 mô hình Logistic Regression và Naive Bayes trên tập kiểm tra, thu được kết quả như sau:

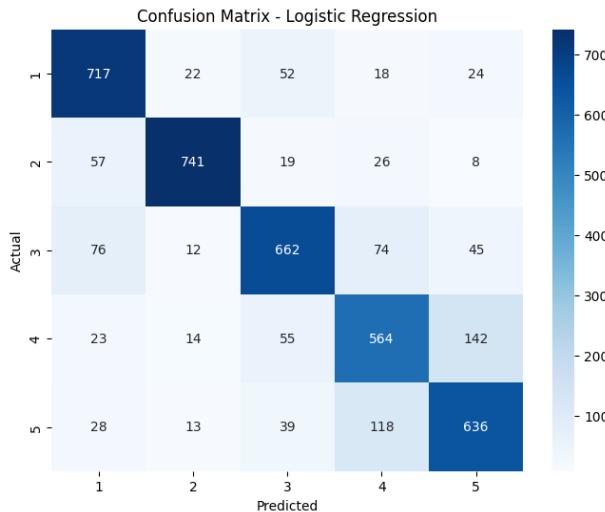
| Classification Report for Logistic Regression model: | | | | |
|--|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 1 | 0.768 | 0.852 | 0.808 | 833 |
| 2 | 0.943 | 0.839 | 0.888 | 851 |
| 3 | 0.785 | 0.757 | 0.771 | 869 |
| 4 | 0.719 | 0.707 | 0.713 | 798 |
| 5 | 0.731 | 0.772 | 0.751 | 834 |
| accuracy | | | 0.786 | 4185 |
| macro avg | 0.789 | 0.785 | 0.786 | 4185 |
| weighted avg | 0.790 | 0.786 | 0.787 | 4185 |

| Classification Report for Naive Bayes model: | | | | |
|--|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 1 | 0.749 | 0.906 | 0.820 | 833 |
| 2 | 1.000 | 0.686 | 0.814 | 851 |
| 3 | 0.875 | 0.695 | 0.775 | 869 |
| 4 | 0.707 | 0.630 | 0.667 | 798 |
| 5 | 0.617 | 0.881 | 0.726 | 834 |
| accuracy | | | 0.760 | 4185 |
| macro avg | 0.790 | 0.760 | 0.760 | 4185 |
| weighted avg | 0.792 | 0.760 | 0.761 | 4185 |

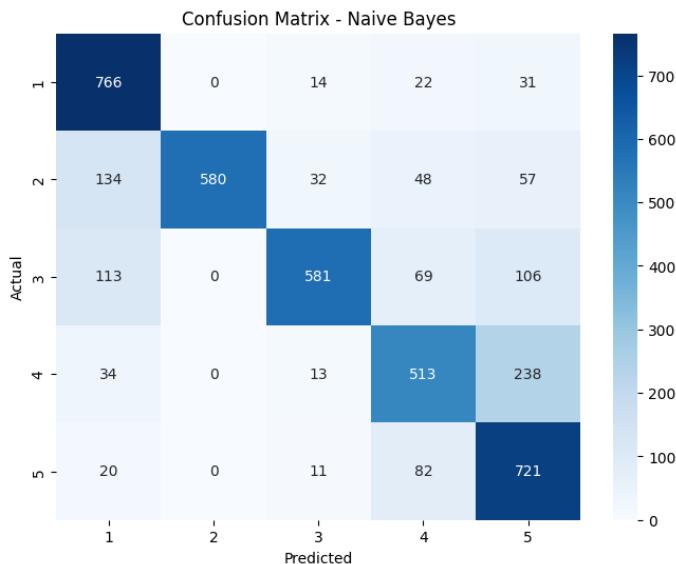
Cả hai mô hình đều có hiệu suất tốt với một số lớp nhất định. Ví dụ, Logistic Regression có precision cao hơn đối với lớp 2, trong khi Naive Bayes có recall cao hơn đối với lớp 1 và lớp 5. Kết quả phụ thuộc vào mục tiêu cụ thể của bạn. Đối với một số lớp, Logistic Regression có hiệu suất tốt hơn, trong khi Naive Bayes làm tốt hơn ở các lớp khác. Nhìn vào các kết quả trên thì kết quả của 2 mô hình Naive Bayes và Logistic

Regression đối với bộ dữ liệu tăng kích thước mẫu khá tốt. Nên tiếp tục phân tích 2 mô hình của bộ dữ liệu này để chọn mô hình tốt nhất.

Sử dụng ma trận nhầm lẫn đánh giá cho 2 mô hình:



Hình 3.2-6 Ma trận nhầm lẫn của mô hình Logistic Regression



Hình 3.2-7 Ma trận nhầm lẫn của mô hình Naive Bayes

Nhìn vào cả hai mô hình, ta thấy biểu đồ của ma trận nhầm lẫn của hai mô hình đều thể hiện được mô hình khá tốt. Tuy nhiên mô hình Naive Bayes đạt hiệu quả hơn trên lớp 1 và 5 so với 3 lớp còn lại, mô hình Logistic Regression có sự chênh lệch giữa các lớp không quá lớn như mô hình Naive Bayes. Và từ kết quả trên, ta có mô hình Logistic Regression có độ chính xác cao hơn, vì thế chọn mô hình này để áp dụng sử dụng và thiết kế giao diện. Ta thực hiện xuất mô hình này ra để sử dụng lại.

3.2.5 Xây dựng Web UI

Lưu mô hình Logistic Regression và lưu bộ dữ liệu cân bằng bằng cách tăng mẫu để sử dụng về sau. Lập hàm ***predict_rate*** dự đoán kết quả đánh giá là bao nhiêu sao, với kết quả tương ứng từ 1 đến 5. Trong hàm này sẽ thực hiện các bước làm sạch dữ liệu, tách dữ liệu và trích xuất đặc trưng.

```

def predict_rate(text):
    text = text_preprocess2(text)
    text = underthesea.word_tokenize(text, format="text")

    balanced_df = pd.read_csv(folder + '/Data/balanced_df.csv')
    X, y = balanced_df['comment_token'], balanced_df['n_star']
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
        # Chia dữ liệu thành tập huấn luyện và tập kiểm tra
        count_vectorizer = CountVectorizer(ngram_range=(1, 5),
stop_words=stopword, max_df=0.5, min_df=5)
        tfidf_vectorizer = TfidfTransformer(use_idf=False,
sublinear_tf = True, norm='l2', smooth_idf=True)

    X_train = count_vectorizer.fit_transform(X_train)
    X_train = tfidf_vectorizer.fit_transform(X_train)

    X_test = count_vectorizer.transform([text])
    X_test = tfidf_vectorizer.transform(X_test)
    # Dự đoán dữ liệu mới
    maxent_model = joblib.load(folder +
'/Output/maxent_model.pkl')
    y_pre = maxent_model.predict(X_test)
    return str(y_pre[0])

```

3.2.5.1 Thư viện Gradio

Thư viện này cho phép xây dựng và triển khai các ứng dụng web. Điểm nổi bật nhất của thư viện này là thực hiện trực quan dữ liệu chỉ với rất ít dòng mã. Một lợi ích khác của thư viện này là làm quá trình thực hiện trở nên nhanh chóng và dễ dàng hơn. Với Gradio, có thể thử nghiệm các đầu vào khác nhau. Việc xác thực mô hình dễ dàng hơn bao giờ hết vì nó có sẵn các liên kết công khai. Do vậy việc triển khai và phân phối các ứng dụng web trở nên rất dễ dàng.

3.2.5.2 Code giao diện

```

def combine(a):
    return predict_rate(a) + '⭐' # Chỗ này điền dự đoán của
mô hình

with gr.Blocks() as demo:

    txt = gr.Textbox(label="Bạn thấy sản phẩm điện thoại này
như thế nào", lines=2)
    txt_2 = gr.Textbox(value="", label="Kết quả đánh giá")
    btn = gr.Button(value="Submit")
    btn.click(combine, inputs=[txt], outputs=[txt_2])

```

```

gr.Markdown("## Đánh giá ví dụ")
gr.Examples(
    ["Tôi thấy sản phẩm này đẹp", "Cấu hình máy mạnh",
    "Máy chạy quá chậm"],
    [txt],
    txt_2,
    combine,
    cache_examples=True)

demo.launch(share=True, server_name="0.0.0.0",
server_port=7860)

```

3.2.5.3 Chức năng của giao diện

Giao diện có chức năng cho phép người dùng nhập vào bình luận về sản phẩm và phân loại bình luận đó được bao nhiêu sao. Nếu người dùng không biết nhập gì có thể nhán ví dụ để xem kết quả phân loại.

- Bước 1: Nhập vào nội dung bình luận.
- Bước 2: Nhấn Submit.
- Bước 3: Chọn ví dụ nếu không biết nhập gì.

3.2.5.4 Thiết kế

Giao diện được thiết kế như sau:

Bạn thấy sản phẩm điện thoại này như thế nào

Kết quả đánh giá

Submit

Đánh giá ví dụ

Examples

Tôi thấy sản phẩm này đẹp

Cấu hình máy mạnh

Máy chạy quá chậm

Hình 3.2-8 Giao diện của Web

3.2.5.5 Chạy thử

Chạy thử đánh giá “Máy chậm, lag, pin yếu” và thu được kết quả như sau:

Bạn thấy sản phẩm điện thoại này như thế nào

Máy chậm, lag, pin yếu

Kết quả đánh giá

2 ★

Submit

Đánh giá ví dụ

≡ Examples

Tôi thấy sản phẩm này đẹp

Cấu hình máy mạnh

Máy chạy quá chậm

Hình 3.2-9 Chạy thử đánh giá "Máy chậm, lag, pin yếu"

3.3 TRIỂN KHAI THỬ NGHIỆM HỆ THỐNG

3.3.1 Các bước cài đặt và triển khai hệ thống

3.3.1.1 Tạo thư mục có chứa đồ án

- Có 2 cách để tạo thư mục.
 - Cách 1: Tạo thư mục mới, đặt tên **python_app** trên máy chủ.
 - Cách 2: Nhập câu lệnh **mkdir** trên Windows Powershell.

PS D:\Docker> mkdir python_app

- Kiểm tra thư mục **python_app** đã có trong máy chủ chưa

| | | |
|-------------------|------------------|-------------|
| Dulieu | 21/05/2024 20:46 | File folder |
| final | 27/05/2024 09:20 | File folder |
| hpc-cluster | 23/05/2024 11:07 | File folder |
| python_app | 27/05/2024 16:11 | File folder |
| Test | 27/05/2024 11:32 | File folder |

Hình 3.3-1 Thư mục đồ án được tạo

- Di chuyển vào trong thư mục đồ án:

Gõ lệnh: cd python_app

PS D:\Docker> cd python_app

3.3.1.2 Thêm các tệp cần thiết đưa vào thư mục

- Sao chép tệp **stopword_train.txt**, đây là danh sách các stopword tổng hợp từ tài liệu tham khảo và từ bộ dữ liệu.
- Sao chép tệp **teencode.xlsx**, đây là bộ từ viết tắt teencode được tổng hợp từ tài liệu tham khảo.
- Sao chép tệp **balanced_df.csv**, đây là bộ dữ liệu UIT-ViSFD sau khi đã thực hiện tiền xử lý và xử lý mất cân bằng.

- Sao chép tệp **maxent_model.pkl**, đây là mô hình máy học Logistic Regression đã được xuất ra.
- Vị trí thư mục **python_app** gồm các tệp sau:

```

    └── python_app/
        ├── balanced_df.csv
        ├── maxent_model.pkl
        ├── stopword_train.txt
        └── teencode.xlsx

```

3.3.1.3 Tạo tệp requirements.txt

- Ý nghĩa của tệp **requirements.txt**:

Tệp requirements.txt chứa danh sách tất cả các thư viện và phiên bản cụ thể mà dự án cần. Điều này giúp đảm bảo rằng tất cả các môi trường phát triển và triển khai đều sử dụng cùng một phiên bản của các thư viện.

- Nội dung của tệp **requirements.txt**

```

underthesea
openpyxl
gradio
joblib

```

3.3.1.4 Tạo file python_app.py

- Tạo file **python_app.py**, có 2 cách để tạo:

- Trong VS Code, chọn **New File ...** rồi đặt tên “**python_app.py**”.
- Nhập câu lệnh **touch python_app.py** trên Windows Powershell.

```
PS D:\Docker\python_app> touch python_app.py
```

- Nhập mã trong file **python_app**:

```

import pandas as pd
import string
import re
import joblib
import matplotlib.pyplot as plt
import underthesea
from underthesea import text_normalize

from sklearn.feature_extraction.text import
CountVectorizer, TfidfTransformer
from sklearn.model_selection import train_test_split
import gradio as gr

def handle_repeated_syllables(text):
    # Sử dụng regex để tìm các từ có âm tiết lặp lại (ví dụ:

```

```

quááááá)
repeated_syllables_pattern = re.compile(r'(\w+?)\1+', re.UNICODE)
# Hàm xử lý việc loại bỏ âm tiết lặp lại
def handle_repetition(match):
    word = match.group(1)
    # Giữ lại chỉ một phần lặp lại và thêm vào từ gốc
    return word

# Áp dụng hàm xử lý vào chuỗi
processed_text =
repeated_syllables_pattern.sub(handle_repetition, text)
return processed_text

EMAIL = re.compile(r"([\w0-9_\.]+)(@)([\d\w\.-]+)(\.)([\w\.]{2,6})")
URL = re.compile(r"https?:\/\/(?!.*\/\/)\S+")
PHONE = re.compile(r"(09|01[2|6|8|9])+([0-9]{9})\b")
MENTION = re.compile(r"@.+?:")
NUMBER = re.compile(r'\b\d+\S*\b')
DATETIME = '\d{1,2}\s?[-]\s?\d{1,2}\s?[-]\s?\d{4}'

# Delete price, 3g/4g/5g
PRICE = r'\b\d{1,4}(?:\.\d{3})*(?:\.\d+)?(?:[ktrđg])\b'

def replace_common_token(txt):
    txt = re.sub(EMAIL, ' ', txt)
    txt = re.sub(URL, ' ', txt)
    txt = re.sub(MENTION, ' ', txt)
    txt = re.sub(DATETIME, ' ', txt)
    txt = re.sub(NUMBER, ' ', txt)
    txt = re.sub(PRICE, ' ', txt)
    return txt

def remove_unnecessary_characters(text):
    RE_CLEAR = re.compile("[\n\r]+")# Thay thế các chuỗi xuống
dòng (\n hoặc \r) bằng một ký tự trắng
    text = re.sub(RE_CLEAR, ' ', text)
    # Sử dụng string.punctuation để lấy tất cả các ký tự dấu
câu
    translator = str.maketrans(' ', ' ', string.punctuation)
    # Loại bỏ dấu câu từ văn bản sử dụng bảng dịch
(translator)
    text = text.translate(translator)

    return text

```

```

def normalize_acronyms(text, teencode_file='teencode.xlsx'):
    # Đọc dữ liệu từ tệp Excel teencode.xlsx
    teencode_df = pd.read_excel(teencode_file, header=None,
                                names=['teencode', 'replace'])

    words = []
    for word in text.strip().split():
        word = word.strip(string.punctuation)
        # Tìm kiếm trong teencode_df và thay thế
        replacement =
            teencode_df.loc[teencode_df['teencode'].str.lower() == word,
                           'replace'].values
        if len(replacement) > 0:
            words.append(replacement[0])
        else:
            words.append(word)

    return ' '.join(words)

stopword = []
with open('stopword_train.txt', 'r', encoding='utf8') as fp:
    for line in fp.readlines():
        stopword.append(line.strip())
len(stopword)

# loại stopword khỏi dữ liệu
def remove_stopwords(line):
    words = []
    for word in line.strip().split():
        if word not in stopword:
            words.append(word)
    return ' '.join(words)

# Không tiến hành tách từ dính nhau vì bộ dữ liệu từ dính nhau
# chỉ được áp dụng cho bộ dữ liệu cũ, không bao quát cho dữ liệu
# mới nhập vào.
def text_preprocess(text):
    # 1. Chuẩn hóa văn bản tiếng việt
    text = text_normalize(text)
    # 2. Xử lý lấy âm tiết
    text = handle_repeated_syllables(text)
    # 3. Loại bỏ các common token
    text = replace_common_token(text)
    # 4. Xóa bỏ dấu câu
    text = remove_unnecessary_characters(text)

```

```

# 5. Đưa về lower
text = text.lower()
# 6. Chuẩn hóa các từ viết tắt cơ bản
text = normalize_acronyms(text)
# 7. Loại bỏ các stopword tiếng Việt
text = remove_stopwords(text)
# 8. Loại bỏ các khoảng trắng liên tiếp
RE_CLEAR = re.compile("\s+")
text = re.sub(RE_CLEAR, ' ', text)
return text

def predict_rate(text):
    text = text_preprocess(text)
    text = underthesea.word_tokenize(text, format="text")

    balanced_df = pd.read_csv('balanced_df.csv')
    X, y = balanced_df['comment_token'], balanced_df['n_star']
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
        # Chia dữ liệu thành tập huấn luyện và tập kiểm tra
        count_vectorizer = CountVectorizer(ngram_range=(1, 5),
stop_words=stopword, max_df=0.5, min_df=5)
        tfidf_vectorizer = TfidfTransformer(use_idf=False,
sublinear_tf = True, norm='l2', smooth_idf=True)

    X_train = count_vectorizer.fit_transform(X_train)
    X_train = tfidf_vectorizer.fit_transform(X_train)

    X_test = count_vectorizer.transform([text])
    X_test = tfidf_vectorizer.transform(X_test)
    # Dự đoán dữ liệu mới
    maxent_model = joblib.load( 'maxent_model.pkl' )
    y_pre = maxent_model.predict(X_test)
    return str(y_pre[0])

# Thiết kế giao diện
def combine(a):
    return predict_rate(a) + ' ★ ' # Chỗ này điền dự đoán của
mô hình

with gr.Blocks() as demo:

    txt = gr.Textbox(label="Bạn thấy sản phẩm điện thoại này
như thế nào", lines=2)
    txt_2 = gr.Textbox(value="", label="Kết quả đánh giá")

```

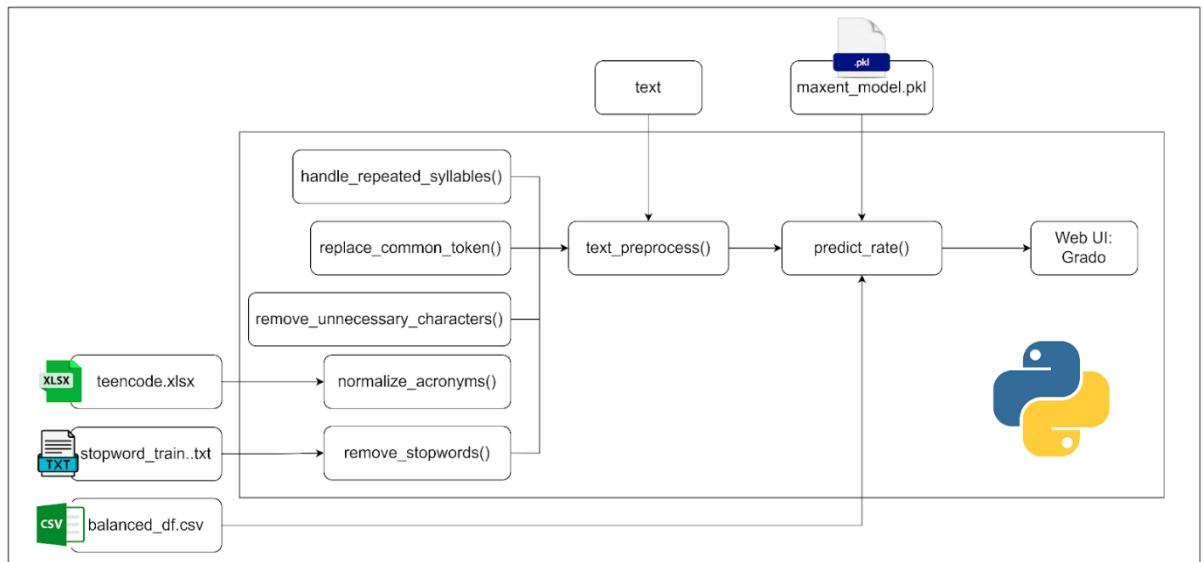
```

btn = gr.Button(value="Submit")
btn.click(combine, inputs=[txt], outputs=[txt_2])
gr.Markdown("## Đánh giá ví dụ")
gr.Examples(
    ["Tôi thấy sản phẩm này đẹp", "Cấu hình máy mạnh",
     "Máy chạy quá chậm"],
    [txt],
    txt_2,
    combine,
    cache_examples=True)

demo.launch(share=True, server_name="0.0.0.0",
            server_port=7860)

```

- Cấu trúc file *python_app*:



Hình 3.3-2 Cấu trúc file *python_app*

- Đoạn mã *python_app.py* được chia thành các phần chính sau:

1. Import các thư viện cần thiết:

Các thư viện được nhập bao gồm các thư viện cho xử lý dữ liệu (`pandas`, `re`), học máy (`sklearn`), xử lý ngôn ngữ tự nhiên (`underthesea`), và tạo giao diện người dùng (`gradio`).

2. Định nghĩa các hàm xử lý văn bản:

- `handle_repeated_syllables`: Xử lý các âm tiết lặp lại trong văn bản.
- `replace_common_token`: Thay thế các token thông dụng như địa chỉ email, URL, số điện thoại, và các số khác bằng khoảng trắng.
- `remove_unnecessary_characters`: Loại bỏ các ký tự không cần thiết như dấu câu và ký tự xuống dòng từ văn bản, ...
- `normalize_acronyms`: Đọc một tệp Excel chứa các từ viết tắt và thay thế các từ viết tắt trong văn bản bằng từ đầy đủ.

- `remove_stopwords`: Loại bỏ các từ không mang nhiều ý nghĩa (stopword) khỏi văn bản dựa trên một danh sách keyword có sẵn.

3. Tiền xử lý văn bản:

Hàm `text_preprocess` kết hợp tất cả các bước xử lý văn bản: chuẩn hóa văn bản, xử lý âm tiết lặp lại, thay thế các token thông dụng, loại bỏ dấu câu, chuyển văn bản về chữ thường, chuẩn hóa các từ viết tắt, loại bỏ các keyword và loại bỏ các khoảng trắng thừa.

4. Dự đoán đánh giá sản phẩm:

Hàm `predict_rate` thực hiện việc tiền xử lý văn bản trước, sau đó tách từ bằng `underthesea`. Sau đó, văn bản được chuyển thành dạng ma trận từ điển (bag-of-words) và áp dụng TF-IDF. Cuối cùng, văn bản được đưa vào mô hình MaxEnt đã được huấn luyện để dự đoán số sao và trả về kết quả.

5. Giao diện người dùng với Gradio:

Một giao diện người dùng được tạo ra bằng `Gradio`, nơi người dùng có thể nhập văn bản đánh giá sản phẩm. Khi nhấn nút "Submit", văn bản sẽ được xử lý và kết quả dự đoán sẽ được hiển thị. Giao diện cũng bao gồm các ví dụ mẫu để người dùng tham khảo.

Vậy luồng công việc như sau:

- Người dùng nhập văn bản bình luận đánh giá sản phẩm qua giao diện Gradio.
- Văn bản đầu vào được làm sạch và chuẩn hóa qua các hàm xử lý.
- Văn bản đã xử lý được đưa vào mô hình học máy để dự đoán số sao.
- Kết quả dự đoán được hiển thị trên giao diện người dùng.

Các bước trên đảm bảo rằng văn bản đầu vào được làm sạch và chuẩn hóa kỹ lưỡng trước khi đưa vào mô hình để đảm bảo độ chính xác của dự đoán.

Lưu ý, dòng cuối cùng của đoạn mã như sau:

```
demo.launch(share=True, server_name="0.0.0.0",
server_port=7860)
```

Điều này sẽ thiết lập Gradio để lắng nghe trên tất cả các giao diện mạng trên cổng 7860, cho phép truy cập từ bên ngoài container Docker, nghĩa là ứng dụng chạy trên 2 URL:

- Local URL (<http://0.0.0.0:7860>):
 - Ứng dụng của bạn đang chạy cục bộ trên máy chủ tại địa chỉ IP 0.0.0.0 và cổng 7860.
 - Địa chỉ IP 0.0.0.0 có nghĩa là ứng dụng đang lắng nghe trên tất cả các giao diện mạng của máy tính.
 - Có thể truy cập ứng dụng bằng cách mở trình duyệt web và nhập địa chỉ <http://localhost:7860> hoặc <http://127.0.0.1:7860>.
- Public URL:

- Đây là một URL công khai được Gradio cung cấp, cho phép người khác truy cập ứng dụng từ bất kỳ đâu trên Internet.
- Gradio là một công cụ tạo giao diện người dùng cho các ứng dụng học máy, và nó cung cấp dịch vụ tạo đường dẫn công khai này để dễ dàng chia sẻ ứng dụng.

3.3.1.5 Tạo Dockerfile

- Tạo Dockerfile, có 2 cách để tạo Dockerfile
 - Trong VS Code, chọn **New File ...** rồi đặt tên "**Dockerfile**".
 - Nhập câu lệnh **touch Dockerfile** trên Windows Powershell.

```
PS D:\Docker\python_app> touch Dockerfile
```

Lưu ý: Tên của Dockerfile phải đúng là "Dockerfile" và không có phần đuôi mở rộng cho loại file này. Nếu không đặt tên đúng, khi build hệ thống sẽ báo lỗi là không tìm thấy file.

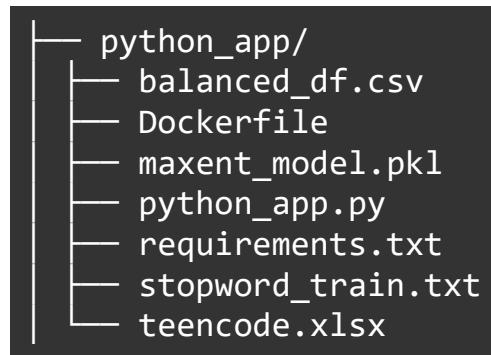
- Khai báo thông số cho Dockerfile
 - Mở Dockerfile vừa tạo lên: có thể mở bằng **note** hoặc **VS Code**, 2 ứng dụng này dễ nhập nội dung, chỉnh sửa và lưu.
 - Nhập nội dung cho Dockerfile:

```
FROM python:3.10
WORKDIR /code
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
CMD [ "python", "python_app.py" ]
```

- Giải thích từng thông số:
 - **FROM python:3.10** chỉ định image python 3.10 được dùng làm image cơ sở trong quá trình build image thực hiện các câu lệnh tiếp theo. Image này được tải về từ Docker Hub.
 - **WORKDIR /code** Đặt thư mục làm việc trong container là **/code**. Tất cả các lệnh tiếp theo sẽ được thực hiện trong thư mục này.
 - **COPY requirements.txt .** Sao chép tệp **requirements.txt** từ máy chủ vào thư mục làm việc **/code** trong container. Tệp này chứa danh sách các thư viện Python cần thiết cho ứng dụng.
 - **RUN pip install --no-cache-dir -r requirements.txt** Chạy lệnh để cài đặt tất cả các thư viện Python được liệt kê trong **requirements.txt**. Tham số **--no-cache-dir** được sử dụng để ngăn pip lưu trữ các gói đã cài đặt trong bộ nhớ đệm, giúp giảm dung lượng của image Docker cuối cùng.
 - **COPY . .** Sao chép toàn bộ nội dung từ thư mục hiện tại của máy chủ vào thư mục làm việc **/code** trong container. Điều này bao gồm

mã nguồn của ứng dụng và bất kỳ tệp nào khác cần thiết cho ứng dụng.

- **CMD ["python", "python_app.py"]** Định nghĩa lệnh mặc định để chạy khi container khởi động. Trong trường hợp này, lệnh này sẽ chạy file python_app.py bằng Python.
- Vậy các tệp trong thư mục **python_app** gồm các tệp sau:



3.3.1.6 Khởi tạo một image Docker từ Dockerfile trong thư mục hiện tại

- Gõ lệnh **docker build -t pythonapp** . trên Windows Powershell.

```
PS D:\Docker\python_app> docker build -t pythonapp .
```

- Giải thích: Dòng lệnh trên là để xây dựng một image Docker từ một Dockerfile có tên là Dockerfile trong thư mục hiện tại (D:\Docker\python_app). Cụ thể:
 - **docker build** Là lệnh để xây dựng một image Docker.
 - **-t pythonapp** Đặt tên cho image được tạo ra là pythonapp.
 - **.** Tham chiếu đến thư mục hiện tại, nơi Dockerfile được đặt.

Như vậy, sau khi lệnh này được thực thi, Docker sẽ đọc Dockerfile từ thư mục hiện tại và sử dụng nó để xây dựng một Docker Image có tên là **pythonapp**.

- Khi khởi tạo thành công, màn hình sẽ hiện như sau:

```
PS D:\Docker\python_app> docker build -t pythonapp .
2024/05/27 16:21:15 http2: server: error reading preface from client //./pipe/docker_engine: file has already been closed
[+] Building 1.6s (10/10) FINISHED
  => [internal] load build definition from Dockerfile
  => => transferring dockerfile: 196B
  => [internal] load metadata for docker.io/library/python:3.10
  => [internal] load .dockerignore
  => => transferring context: 2B
  => [1/5] FROM docker.io/library/python:3.10@sha256:817c0d8684087acb6d88f0f0951f9a541aa3e762302aa5e8f439d5d12edd48ad
  => [internal] load build context
  => => transferring context: 5.98kB
  => CACHED [2/5] WORKDIR /code
  => CACHED [3/5] COPY requirements.txt .
  => CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt
  => [5/5] COPY . .
  => => exporting to image
  => => exporting layers
  => => writing image sha256:0aa0c979463bcbd91eeddfe6c01e4d246aba093a615b040dbbd6e88f8be593aa
  => => naming to docker.io/library/pythonapp

View build details: docker-desktop://dashboard/build/default/imr9pe5m3uxv37uro6n68ptse
What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
```

Hình 3.3-3 Khởi tạo image pythonapp thành công

- Kiểm tra đã có Image **pythonapp** chưa:

- Gõ lệnh **docker images** trên Windows Powershell.

```
PS D:\Docker\python_app> docker images
REPOSITORY      TAG      IMAGE ID      CREATED       SIZE
pythonapp        latest   0aa0c979463b  36 hours ago  1.65GB
ubuntu          latest   bf3dc08bfed0  4 weeks ago   76.2MB
busybox          latest   65ad0d468eb1  12 months ago  4.26MB
hello-world     latest   d2c94e258dcb  13 months ago  13.3kB
```

Hình 3.3-4 Kiểm tra các Image trong Windows PowerShell

- Hoặc kiểm tra trong Docker.

| Name | Tag | Status | Created | Size | Actions |
|---|--------|--------|-----------|---------|---------|
| pythonapp 0aa0c979463b | latest | In use | 1 day ago | 1.65 GB | ▶ ⋮ |

Hình 3.3-5 Kiểm tra các Image trong Docker

3.3.1.7 Khởi tạo một container Docker từ image pythonapp

- Gõ lệnh **docker run -it -p 7860:7860 pythonapp** trên Windows Powershell.

```
PS D:\Docker\python_app> docker run -it -p 7860:7860 pythonapp
```

- Giải thích: Dòng lệnh này để chạy một container Docker từ image **pythonapp** đã được tạo ra trước đó. Cụ thể:
 - **docker run**: Lệnh để chạy một container từ một image Docker.
 - **-it**: Tùy chọn này kết hợp hai tham số:
 - **-i** (interactive): Giữ stdin mở ngay cả khi không đính kèm.
 - **-t** (tty): Gán một terminal ảo để bạn có thể tương tác với container.
 - **-p 7860:7860**: Chuyển tiếp cổng, nghĩa là ánh xạ cổng 7860 trên máy chủ (host) với cổng 7860 bên trong container. Điều này cho phép truy cập vào dịch vụ đang chạy trên cổng 7860 bên trong container thông qua cổng 7860 trên máy chủ.
 - **pythonapp**: Tên của image muốn sử dụng để tạo và chạy container.

Như vậy, sau khi lệnh này được thực thi, Docker sẽ đọc Dockerfile từ thư mục hiện tại và sử dụng nó để xây dựng một Docker Image có tên là **pythonapp**.

- Khi khởi tạo thành công, màn hình sẽ hiện như sau:

```

PS D:\Docker\python_app> docker run -it -p 7860:7860 pythonapp
Caching examples at: '/code/gradio_cached_examples/5'
Caching example 1/3
/usr/local/lib/python3.10/site-packages/sklearn/base.py:376: InconsistentVersionWarning: Trying to unpickle estimator LogisticRegression from version 1.2.2 when using version 1.5.0. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
    warnings.warn(
Caching example 2/3
/usr/local/lib/python3.10/site-packages/sklearn/base.py:376: InconsistentVersionWarning: Trying to unpickle estimator LogisticRegression from version 1.2.2 when using version 1.5.0. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
    warnings.warn(
Caching example 3/3
/usr/local/lib/python3.10/site-packages/sklearn/base.py:376: InconsistentVersionWarning: Trying to unpickle estimator LogisticRegression from version 1.2.2 when using version 1.5.0. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
    warnings.warn(
Running on local URL: http://0.0.0.0:7860
Running on public URL: https://9950d45d07602283f0.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Terminal to deploy to Spaces (https://huggingface.co/spaces)

```

Hình 3.3-6 Khởi chạy Container sử dụng Image pythonapp

- Đặc biệt quan tâm 2 dòng dưới đây:

```

Running on local URL: http://0.0.0.0:7860
Running on public URL: https://9950d45d07602283f0.gradio.live

```

Hình 3.3-7 Các đường link truy cập web

Vậy có thể chạy ứng dụng trên 2 URL:

- Local URL (<http://0.0.0.0:7860>): Truy cập ứng dụng bằng cách mở trình duyệt web và nhập địa chỉ <http://localhost:7860> hoặc <http://127.0.0.1:7860>.
- Public URL: Truy cập ứng dụng bằng cách mở trình duyệt web và nhập địa chỉ <https://9950d45d07602283f0.gradio.live>. Link được share trong vòng 72 tiếng.
- Kiểm tra trong Docker đã có container chưa:
 - Gõ lệnh **docker ps** trên Windows Powershell.

```

PS D:\Docker\python_app> docker ps

```

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
|--------------|-----------|------------------------|-------------------|---------------|------------------------|----------------|
| 1356b47b4ef8 | pythonapp | "python python_app.py" | About an hour ago | Up 25 seconds | 0.0.0.0:7860->7860/tcp | hopeful_jepsen |

Hình 3.3-8 Kiểm tra các Container đang chạy trong Windows PowerShell

- Hoặc kiểm tra trong Docker và thấy Container đang chạy:

| Name | Image | Status | CPU (%) | Port(s) | Last started | Actions |
|---|-----------|---------|---------|---------------------------|----------------|---|
|  festive_may e8f34962054c | pythonapp | Running | 0.66% | 7860:7860 | 10 minutes ago |  |

Hình 3.3-9 Kiểm tra các Container đang chạy trong Docker

3.3.2 Tiết kiệm truy cập và chạy thử nghiệm

3.3.2.1 Truy cập và chạy thử nghiệm trên local URL

- Mở trình duyệt lên và gõ link miền trang chủ localhost: <http://localhost:7860> hoặc <http://127.0.0.1:7860>.

- Màn hình sẽ hiển thị như hình bên dưới.

Bạn thấy sản phẩm điện thoại này như thế nào?

Kết quả đánh giá

Submit

Đánh giá ví dụ

Examples

Tôi thấy sản phẩm này đẹp Cấu hình máy mạnh Máy chạy quá chậm

Use via API · Built with Gradio

Hình 3.3-10 Giao diện Web khi truy cập Local URL

- Nhập vào nội dung:

- Nhập bình luận toàn khen: “*Chiếc điện thoại này thật sự tuyệt vời! Màn hình sắc nét, hiệu suất mượt mà và thời lượng pin ấn tượng. Camera chụp ảnh đẹp, đặc biệt là chụp đêm. Thiết kế sang trọng, cầm rất chắc tay. Rất hài lòng với sản phẩm này, đáng đồng tiền bát gạo!*”.
 - Üng dụng mất **3.2s** để chạy.
 - Đánh giá 5 sao.

Bạn thấy sản phẩm điện thoại này như thế nào?

Chiếc điện thoại này thật sự tuyệt vời! Màn hình sắc nét, hiệu suất mượt mà và thời lượng pin ấn tượng. Camera chụp ảnh đẹp, đặc biệt là chụp đêm. Thiết kế sang trọng, cầm rất chắc tay. Rất hài lòng với sản phẩm này, đáng đồng tiền bát gạo!

Kết quả đánh giá

5★

Submit

Đánh giá ví dụ

Examples

Tôi thấy sản phẩm này đẹp Cấu hình máy mạnh Máy chạy quá chậm

Hình 3.3-11 Chạy thử nghiệm với một bình luận toàn khen

- Nhập bình luận vừa khen vừa chê: “*Điện thoại này có camera chụp ảnh đẹp, thiết kế sang trọng và hiệu suất ổn định. Tuy nhiên, pin hao nhanh và giao diện hơi khó dùng lúc đầu. Tổng thể là sản phẩm tốt nhưng có vài điểm cần cải thiện.*”.
 - Üng dụng mất **3s** để chạy.
 - Đánh giá 4 sao.

Bạn thấy sản phẩm điện thoại này như thế nào

Điện thoại này có camera chụp ảnh đẹp, thiết kế sang trọng và hiệu suất ổn định. Tuy nhiên, pin hao nhanh và giao diện hơi khó dùng lúc đầu. Tổng thể là sản phẩm tốt nhưng có vài điểm cần cải thiện.

Kết quả đánh giá

4 ★

Submit

Đánh giá ví dụ

≡ Examples

Tôi thấy sản phẩm này đẹp

Cấu hình máy mạnh

Máy chạy quá chậm

Hình 3.3-12 Chạy thử nghiệm với một bình luận vừa khen vừa chê

- Nhập bình luận toàn chê: “*Thất vọng với điện thoại này. Pin kém, màn hình mờ, hay giật lag và camera chụp tệ trong điều kiện thiếu sáng. Thiết kế không chắc tay và dễ bám vân tay. Không đáng giá tiền.*”.
 - Ứng dụng mất 3s để chạy.
 - Đánh giá 1 sao.

Bạn thấy sản phẩm điện thoại này như thế nào

Thất vọng với điện thoại này. Pin kém, màn hình mờ, hay giật lag và camera chụp tệ trong điều kiện thiếu sáng. Thiết kế không chắc tay và dễ bám vân tay. Không đáng giá tiền.

Kết quả đánh giá

1 ★

Submit

Đánh giá ví dụ

≡ Examples

Tôi thấy sản phẩm này đẹp

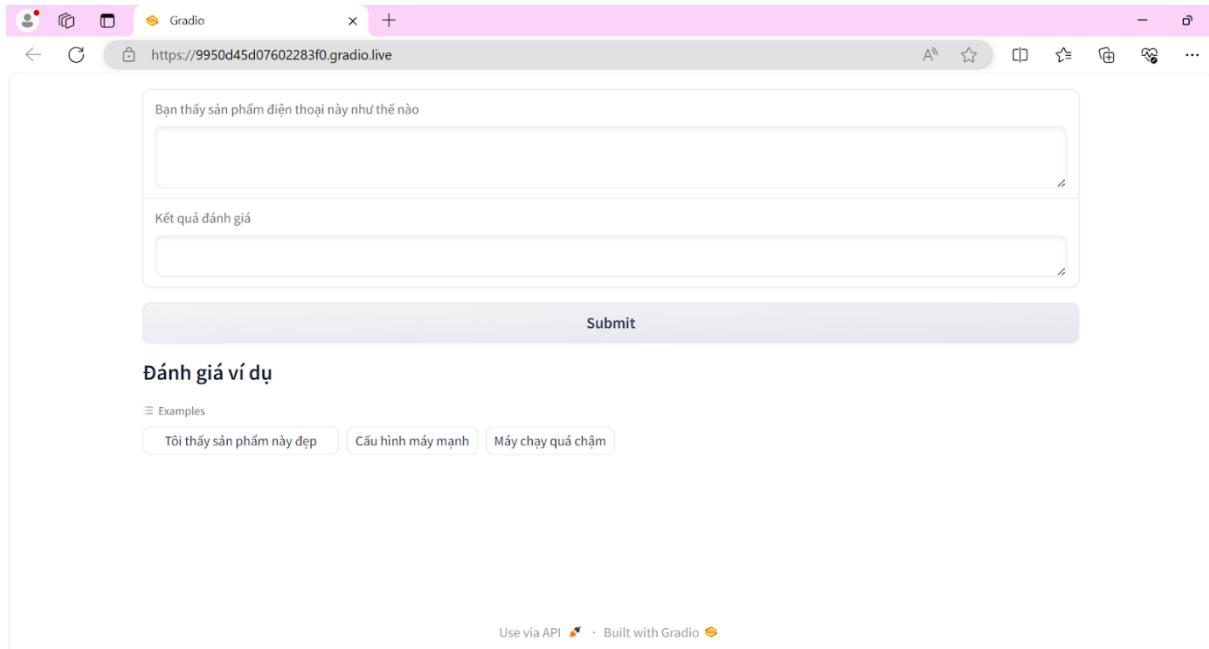
Cấu hình máy mạnh

Máy chạy quá chậm

Hình 3.3-13 Chạy thử nghiệm với bình luận toàn chê

3.3.2.2 Truy cập và chạy thử nghiệm trên public URL

- Mở trình duyệt lên và gõ link <https://9950d45d07602283f0.gradio.live>.
- Màn hình sẽ hiển thị như hình bên dưới.



Hình 3.3-14 Giao diện web khi truy cập public URL

- Nhập vào nội dung:

- Nhập bình luận toàn khen: “*Chiếc điện thoại này thật sự tuyệt vời! Màn hình sắc nét, hiệu suất mượt mà và thời lượng pin ấn tượng. Camera chụp ảnh đẹp, đặc biệt là chụp đêm. Thiết kế sang trọng, cầm rất chắc tay. Rất hài lòng với sản phẩm này, đáng đồng tiền bát gạo!*”.
 - Ứng dụng mất **4.8s** để chạy.
 - Đánh giá 5 sao.

Hình 3.3-15 Chạy thử nghiệm với bình luận toàn khen

- Nhập bình luận vừa khen vừa chê: “*Điện thoại này có camera chụp ảnh đẹp, thiết kế sang trọng và hiệu suất ổn định. Tuy nhiên, pin hao nhanh và giao diện hơi khó dùng lúc đầu. Tổng thể là sản phẩm tốt nhưng có vài điểm cần cải thiện.*”.
 - Ứng dụng mất **3.8s** để chạy.
 - Đánh giá 4 sao.

Bạn thấy sản phẩm điện thoại này như thế nào

Điện thoại này có camera chụp ảnh đẹp, thiết kế sang trọng và hiệu suất ổn định. Tuy nhiên, pin hao nhanh và giao diện hơi khó dùng lúc đầu. Tổng thể là sản phẩm tốt nhưng có vài điểm cần cải thiện.

Kết quả đánh giá

4 ★

Submit

Đánh giá ví dụ

≡ Examples

Tôi thấy sản phẩm này đẹp

Cấu hình máy mạnh

Máy chạy quá chậm

Hình 3.3-16 Chạy thử nghiệm với bình luận vừa khen vừa chê

- Nhập bình luận toàn chê: “*Thất vọng với điện thoại này. Pin kém, màn hình mờ, hay giật lag và camera chụp tệ trong điều kiện thiếu sáng. Thiết kế không chắc tay và dễ bám vân tay. Không đáng giá tiền.*”.
 - Ứng dụng mất **3.8s** để chạy.
 - Đánh giá 1 sao.

Bạn thấy sản phẩm điện thoại này như thế nào

Thất vọng với điện thoại này. Pin kém, màn hình mờ, hay giật lag và camera chụp tệ trong điều kiện thiếu sáng. Thiết kế không chắc tay và dễ bám vân tay. Không đáng giá tiền.

Kết quả đánh giá

1 ★

Submit

Đánh giá ví dụ

≡ Examples

Tôi thấy sản phẩm này đẹp

Cấu hình máy mạnh

Máy chạy quá chậm

Hình 3.3-17 Chạy thử nghiệm với bình luận toàn chê

3.3.3 Tiến hành thử nghiệm khi tắt Container và chạy lại Container

3.3.3.1 Thủ nghiệm khi tắt container

Tiến hành thử nghiệm khi stop container thì trang web có còn hoạt động hay không.

- Tắt hoạt động container vừa tạo.
 - Gõ lệnh **docker stop 135** trên Windows Powershell.

```
PS D:\Docker\python_app> docker stop 135
```

- Gõ lệnh **docker ps** trên Windows Powershell để kiểm tra container đã dừng chưa.

```
PS D:\Docker\python_app> docker ps
```

| | | | | | | | |
|------------------------------------|--------------|-------|---------|---------|--------|-------|-------|
| PS D:\Docker\python_app> docker ps | CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
|------------------------------------|--------------|-------|---------|---------|--------|-------|-------|

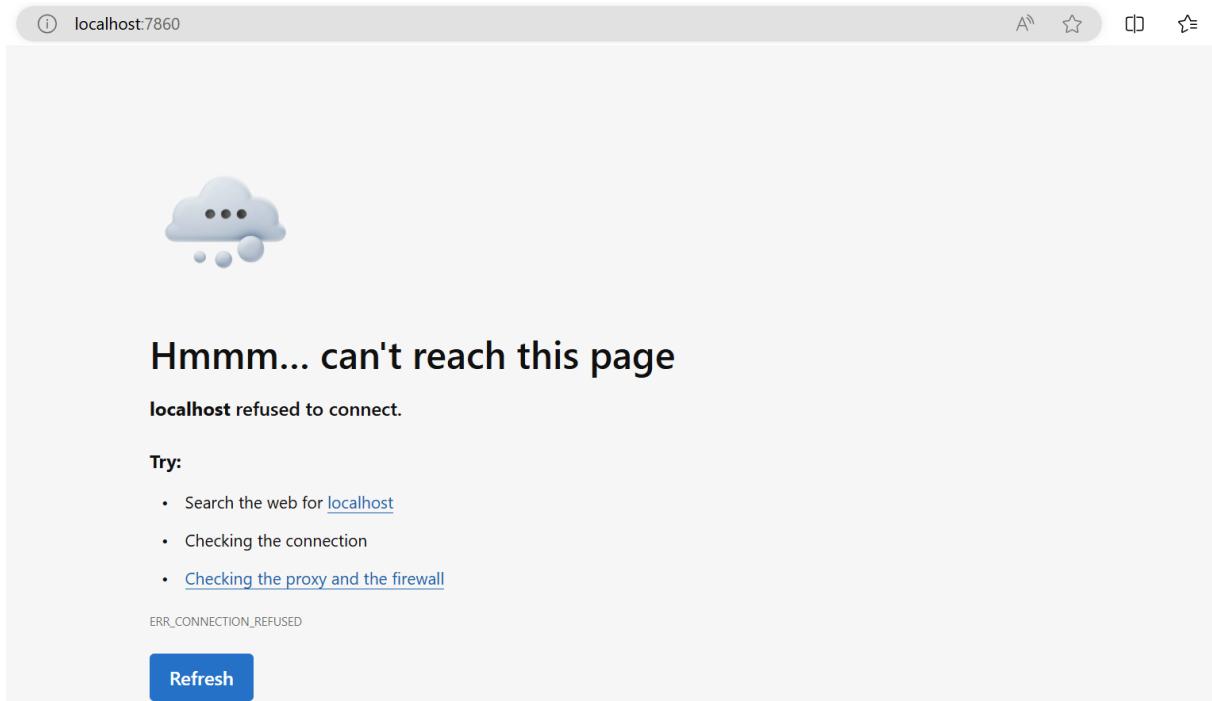
Hình 3.3-18 Kiểm tra các Container đang chạy trong Windows PowerShell

- Hoặc kiểm tra trong Docker.

| Name | Image | Status | CPU (%) | Port(s) | Last started | Actions |
|---|---------------------------|--------------|---------|-----------|---------------|---|
|  hopeful_jep 1356b47b4ef8 | pythonapp | Exited (137) | N/A | 7860:7860 | 7 minutes ago |  |

Hình 3.3-19 Kiểm tra các Container đang chạy trong Docker

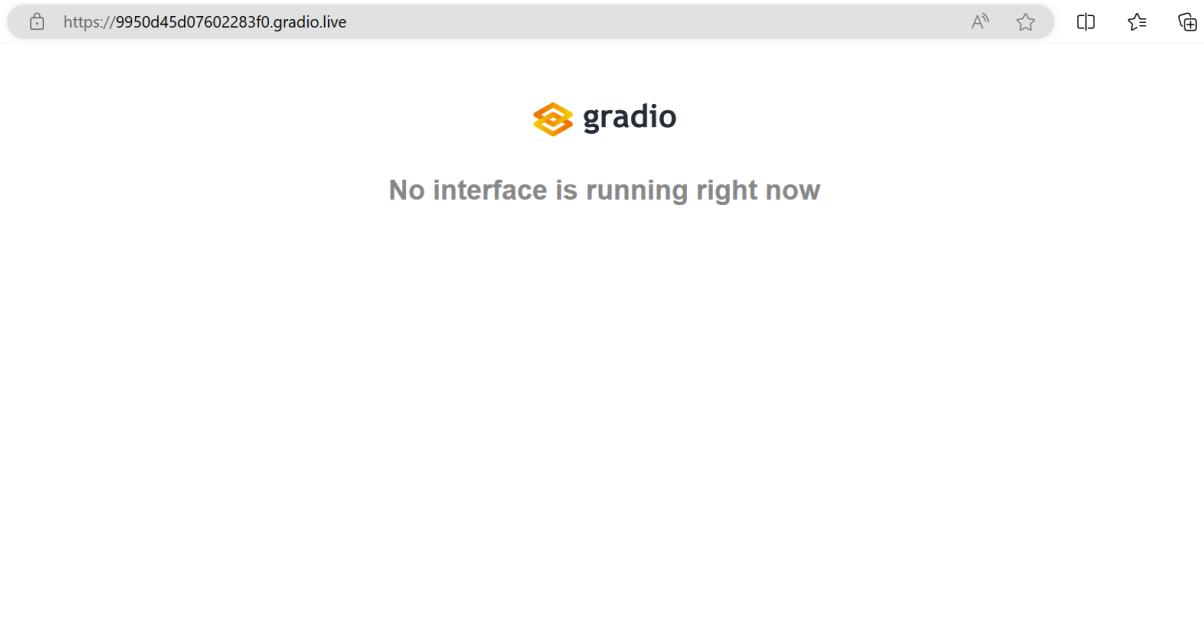
- Truy cập và chạy thử nghiệm trên local URL.
 - Mở trình duyệt lên và gõ link miền trang chủ localhost: <http://localhost:7860> hoặc <http://127.0.0.1:7860>.
 - Màn hình hiển thị như hình bên dưới.



Hình 3.3-20 Không truy cập được Local URL

Khi tắt container Docker, dịch vụ web đang chạy bên trong container cũng sẽ bị tắt. Điều này dẫn đến việc không thể truy cập vào URL như <http://localhost:7860> vì ứng dụng không còn chạy nữa.

- Truy cập và chạy thử nghiệm trên public URL.
 - Mở trình duyệt lên và gõ link <https://9950d45d07602283f0.gradio.live>.
 - Màn hình hiển thị như hình bên dưới.



Hình 3.3-21 Không truy cập được Public URL

Khi tắt hoặc dừng container Docker, tất cả các dịch vụ bên trong container cũng bị dừng, bao gồm cả ứng dụng Gradio đang chạy. URL công khai không còn phục vụ bất kỳ nội dung nào vì không có ứng dụng nào đang chạy, điều này khiến cho URL công khai của Gradio hiển thị thông báo "No interface is running right now" vì không có ứng dụng nào đang chạy để cung cấp giao diện người dùng. Khi container dừng, URL này không còn liên kết tới bất kỳ dịch vụ nào đang hoạt động, dẫn đến thông báo như trên hình.

3.3.3.2 Thủ nghiệm chạy lại container đã bị dừng

Tiến hành thử nghiệm chạy lại container đã dừng để kiểm tra các link trang web có còn hoạt động hay không.

- Tắt hoạt động container vừa tạo.
 - Gõ lệnh **docker start 135** trên Windows Powershell.

```
PS D:\Docker\python_app> docker start 135
```

- Gõ lệnh **docker ps** trên Windows Powershell để kiểm tra container đã dừng chưa.

```
PS D:\Docker\python_app> docker ps
```

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
|--------------|-----------|------------------------|-------------------|---------------|----------------------|----------------|
| 1356b47b4ef8 | pythonapp | "python python_app.py" | About an hour ago | Up 25 seconds | 0.0.0:7860->7860/tcp | hopeful_jepsen |

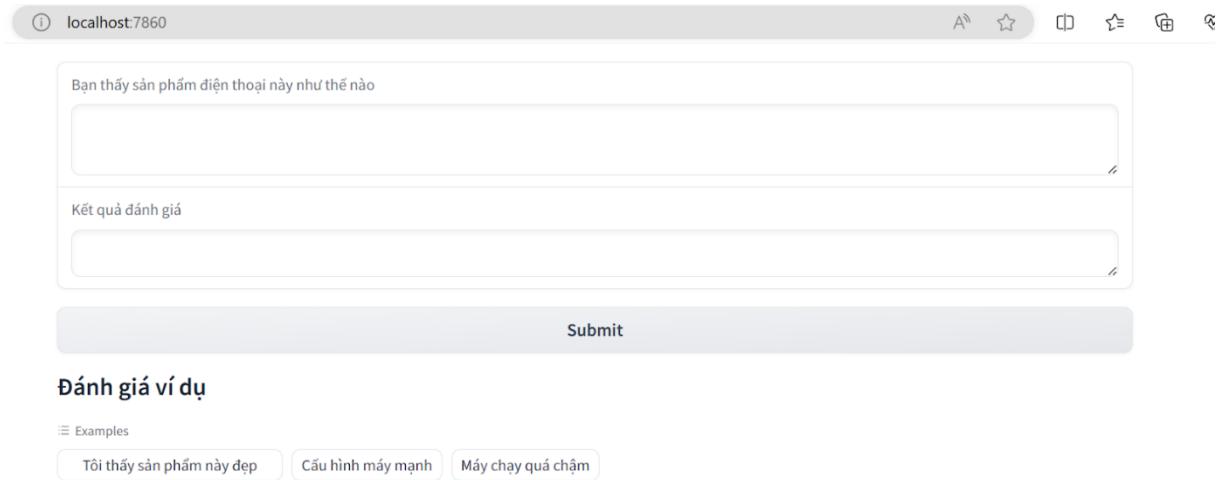
Hình 3.3-22 Kiểm tra các Container đang chạy trong Windows PowerShell

- Hoặc kiểm tra trong Docker.

| Name | Image | Status | CPU (%) | Port(s) | Last started | Actions |
|--|---------------------------|---------|---------|---------------------------|---------------|---|
|  hopeful_jep 1356b47b4efc | pythonapp | Running | N/A | 7860:7860 | 0 seconds ago |    |

Hình 3.3-23 Kiểm tra các Container đang chạy trong Docker

- Truy cập và chạy thử nghiệm trên local URL.
 - Mở trình duyệt lên và gõ link miền trang chủ localhost: <http://localhost:7860> hoặc <http://127.0.0.1:7860>.
 - Màn hình hiển thị như hình bên dưới.



localhost:7860

Bạn thấy sản phẩm điện thoại này như thế nào?

Kết quả đánh giá

Submit

Đánh giá ví dụ

⋮ Examples

Tôi thấy sản phẩm này đẹp Cấu hình máy mạnh Máy chạy quá chậm

Hình 3.3-24 Truy cập được Local URL

Khi tắt và chạy lại container Docker, local URL vẫn hoạt động bình thường vì container đã được cấu hình để chứa ứng dụng web và lắng nghe trên cổng 7860. Lúc chạy container bằng lệnh `docker run -it -p 7860:7860 pythonapp`, đã định cấu hình chuyển tiếp cổng 7860 của máy chủ local sang cổng 7860 trong container, cho phép truy cập ứng dụng web của bạn thông qua local URL. Khi container được khởi động lại, quá trình này không thay đổi, do đó local URL vẫn có thể truy cập được.

- Truy cập và chạy thử nghiệm trên public URL.
 - Mở trình duyệt lên và gõ link <https://9950d45d07602283f0.gradio.live>.
 - Màn hình hiển thị như hình bên dưới.



Hình 3.3-25 Không truy cập được Public URL

Khi sử dụng dịch vụ như Gradio để tạo public URL, dịch vụ này thường cung cấp một URL public để truy cập ứng dụng thông qua internet. Tuy nhiên, mỗi khi khởi động lại container hoặc dịch vụ Gradio, URL public này có thể thay đổi. Lý do cho việc thay đổi URL public có thể là do cơ chế cấp phát địa chỉ IP và cổng của dịch vụ. Khi khởi động lại container hoặc dịch vụ, cơ chế này có thể quyết định cấp phát một địa chỉ IP và cổng mới cho ứng dụng, dẫn đến việc URL public cũ không còn hợp lệ nữa. Do đó, nếu gặp thông báo "No interface is running" khi truy cập public URL của ứng dụng sau khi khởi động lại container, có thể là do URL public đã thay đổi. Trong trường hợp này, cần kiểm tra lại URL mới được cung cấp bởi dịch vụ Gradio sau khi khởi động lại container hoặc dịch vụ.

Để kiểm tra URL mới được cung cấp sau khi khởi động lại container hoặc dịch vụ Gradio, kiểm tra log khi khởi động lại container hoặc dịch vụ. Trong quá trình khởi động lại container hoặc dịch vụ Gradio, thông tin về URL mới có thể được ghi lại trong log.

- Gõ lệnh **docker logs 135** trên Windows Powershell để xem log của container Docker và tìm kiếm thông tin về URL mới.

```
PS D:\Docker\python_app> docker logs 135
```

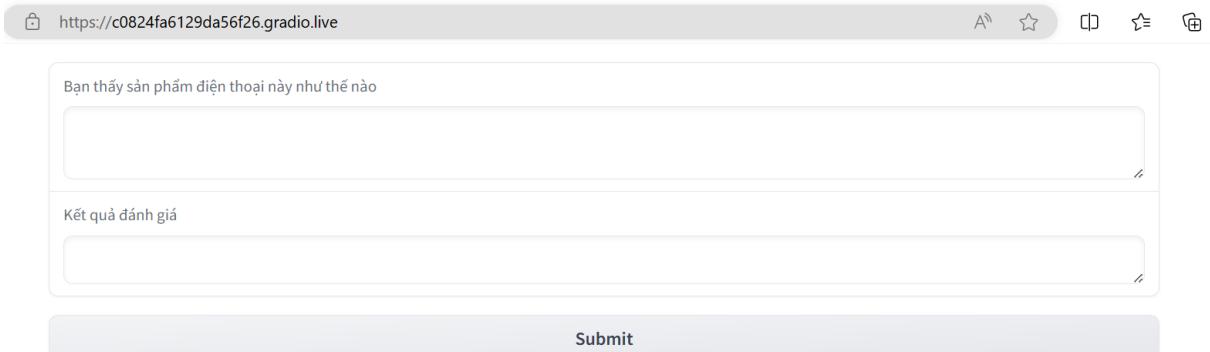
```

PS D:\Docker\python_app> docker logs 135
Caching examples at: '/code/gradio_cached_examples/5'
Caching example 1/3
/usr/local/lib/python3.10/site-packages/sklearn/base.py:376: InconsistentVersionWarning: Trying to unpickle estimator LogisticRegression fr
om version 1.2.2 when using version 1.5.0. This might lead to breaking code or invalid results. Use at your own risk. For more info please
refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
Caching example 2/3
/usr/local/lib/python3.10/site-packages/sklearn/base.py:376: InconsistentVersionWarning: Trying to unpickle estimator LogisticRegression fr
om version 1.2.2 when using version 1.5.0. This might lead to breaking code or invalid results. Use at your own risk. For more info please
refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
Caching example 3/3
/usr/local/lib/python3.10/site-packages/sklearn/base.py:376: InconsistentVersionWarning: Trying to unpickle estimator LogisticRegression fr
om version 1.2.2 when using version 1.5.0. This might lead to breaking code or invalid results. Use at your own risk. For more info please
refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
Running on local URL: http://0.0.0:7860
Running on public URL: https://9950d45d07602283f0.gradio.live

```

Hình 3.3-26 Thông tin Container khi chạy lại

- Mở trình duyệt lên và gõ link <https://c0824fa6129da56f26.gradio.live>.
- Màn hình hiển thị như hình bên dưới.



Đánh giá ví dụ

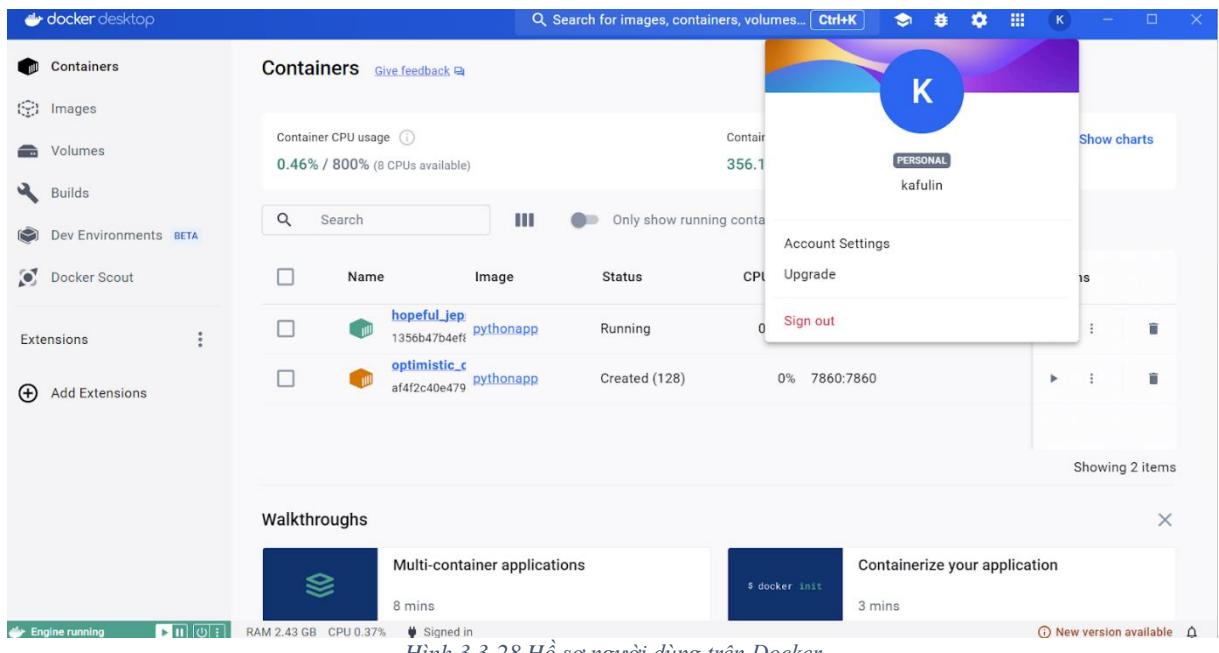
Examples
 Tôi thấy sản phẩm này đẹp Cấu hình máy mạnh Máy chạy quá chậm

Hình 3.3-27 Truy cập được Public URL

3.3.4 Đẩy Docker Imgae lên Docker Hub

3.3.4.1 Kiểm tra username trong Docker

- Mở Docker, nhập vào biểu tượng hồ sơ hoặc tên ở góc trên bên phải của trang.



Hình 3.3-28 Hồ sơ người dùng trên Docker

- Username là **kafulin**.

3.3.4.2 Đăng nhập vào Docker Hub

- Nhập câu lệnh **docker login** trên Windows Powershell.

```
PS D:\Docker> docker login
PS D:\Docker\python_app> docker login
Authenticating with existing credentials...
Login Succeeded
```

Hình 3.3-29 Màn hình khi login Docker thành công

3.3.4.3 Tag image để chuẩn bị đẩy lên Docker Hub

- Kiểm tra ID của image cần sử dụng. Nhập câu lệnh **docker image** trên Windows Powershell.

```
PS D:\Docker> docker image
PS D:\Docker\python_app> docker images
REPOSITORY      TAG        IMAGE ID      CREATED       SIZE
pythonapp        latest     0aa0c979463b  38 hours ago  1.65GB
ubuntu           latest     bf3dc08bfed0  4 weeks ago   76.2MB
busybox          latest     65ad0d468eb1  12 months ago  4.26MB
hello-world      latest     d2c94e258dcb  13 months ago  13.3kB
```

Hình 3.3-30 Kiểm tra các Image trong Windows PowerShell

Nếu image chưa được gắn tag, hãy gắn tag cho nó bằng lệnh: **docker tag**

- Nhập câu lệnh **docker tag 0aa0c979463b kafulin/pythonapp:latest** trên Windows Powershell.

```
PS D:\Docker> docker tag 0aa0c979463b kafulin/pythonapp:latest
```

3.3.4.4 Đẩy image lên Docker Hub

- Nhập câu lệnh `docker tag 0aa0c979463b kafulin/pythonapp:latest` trên Windows Powershell.

```
PS D:\Docker> docker push kafulin/pythonapp:latest
```

```
PS D:\Docker\python_app> docker push kafulin/pythonapp:latest
The push refers to repository [docker.io/kafulin/pythonapp]
1c49d8afe0fd: Pushed
3c9943024e23: Pushed
1df735381fa7: Pushed
f12a8d06148a: Pushed
b43c1d914be6: Mounted from library/python
0a1f19c0d90d: Mounted from library/python
a8273e3b1197: Mounted from library/python
cbe4fb5e267b: Mounted from library/python
734c0f0b65c2: Mounted from library/node
8845ab872c1c: Mounted from library/node
d7d4c2f9d26b: Mounted from library/node
bbe1a212f7e9: Mounted from library/node
latest: digest: sha256:e7b475580f30ee2f81ea2014720f91625f76c605f3e7992dbae66a19faa09b8a size: 2844
```

Hình 3.3-31 Đẩy Image lên Docker Hub

- Kiểm tra image đã được đẩy lên trên Docker Hub chưa.

1. Truy cập Docker Hub

- Mở trình duyệt và truy cập Docker Hub.
- Đăng nhập vào tài khoản Docker Hub của bạn bằng cách sử dụng địa chỉ email và mật khẩu đã đăng ký.

2. Kiểm tra Hồ Sơ Người Dùng (Profile)

- Sau khi đăng nhập, nhập vào biểu tượng hồ sơ hoặc tên của bạn ở góc trên bên phải của trang.
- Từ menu thả xuống, chọn "My Profile".
- Image pythonapp đã có mặt trên Docker Hub. Truy cập vào link sau để mở: [kafulin/pythonapp - Docker Image | Docker Hub](#)

Lê Trần Khánh Phú [Edit profile](#)
Community User Joined April 8, 2024

[Repositories](#) [Starred](#) [Contributed](#)

Displaying 1 to 1 repositories

| | | |
|--|--|-------|
| | kafulin/pythonapp By kafulin • Updated 10 minutes ago | 1 · 0 |
|--|--|-------|

Hình 3.3-32 Image pythonapp đã được đưa lên Docker Hub

3.3.4.5 Thủ tục tải Image từ Docker Hub

- Nhập câu lệnh **docker pull kafulin/pythonapp** trên Windows Powershell.

```
PS D:\Docker> docker pull kafulin/pythonapp
PS D:\Docker\python_app> docker pull kafulin/pythonapp
Using default tag: latest
latest: Pulling from kafulin/pythonapp
c6cf28de8a06: Pull complete
891494355808: Pull complete
6582c62583ef: Pull complete
bf2c3e352f3d: Pull complete
a99509a32390: Pull complete
946285778af4: Pull complete
b2ab5d29389b: Pull complete
d76e704b1be9: Pull complete
c757e6dc09b0: Pull complete
70e301755c28: Pull complete
a9b269a8ea17: Pull complete
be968585335b: Pull complete
Digest: sha256:e7b475580f30ee2f81ea2014720f91625f76c605f3e7992dbae66a19faa09b8a
Status: Downloaded newer image for kafulin/pythonapp:latest
docker.io/kafulin/pythonapp:latest

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview kafulin/pythonapp
```

Hình 3.3-33 Tải image pythonapp về

- Kiểm tra image có trong Docker chưa:

```
PS D:\Docker\python_app> docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
kafulin/pythonapp   latest   0aa0c979463b   39 hours ago  1.65GB
```

Hình 3.3-34 Kiểm tra các Image trong Windows PowerShell

Chương 4. BÀN LUẬN VÀ ĐÁNH GIÁ

4.1 BÀN LUẬN VÀ ĐÁNH GIÁ TRIỂN KHAI PYTHON TRÊN DOCKER IMAGE OFFICIAL

1. Lợi ích của việc sử dụng Docker Official Image cho Python

- Đồng nhất môi trường:
 - Docker Official Image cho Python giúp đảm bảo môi trường triển khai đồng nhất giữa các máy chủ phát triển, kiểm thử và sản xuất.
 - Tránh được các vấn đề phát sinh do sự khác biệt về môi trường, phiên bản phần mềm, hoặc cấu hình hệ thống.
- Dễ dàng quản lý và triển khai:
 - Dockerfile của bạn đơn giản và rõ ràng, chỉ cần định nghĩa các bước cần thiết để cài đặt các gói yêu cầu và chạy ứng dụng.
 - Việc sử dụng Docker Compose giúp quản lý và triển khai các dịch vụ liên quan một cách dễ dàng.
- Cộng đồng hỗ trợ lớn:
 - Docker Official Image cho Python được duy trì bởi cộng đồng lớn và đáng tin cậy, đảm bảo các bản cập nhật bảo mật và vá lỗi nhanh chóng.
 - Tài liệu phong phú và sự hỗ trợ từ cộng đồng giúp giải quyết các vấn đề phát sinh dễ dàng hơn.

2. Hiệu năng và tài nguyên

- Tối ưu hóa hiệu năng:
 - Việc sử dụng Docker Image Official có thể không tối ưu hoàn toàn cho một số ứng dụng cần yêu cầu cao về hiệu năng hoặc tài nguyên cụ thể. Tuy nhiên, cho các ứng dụng thông thường, đây vẫn là lựa chọn tốt.
 - Các lớp cache của Docker giúp cải thiện hiệu năng build và giảm thời gian khởi động container.
- Quản lý tài nguyên:
 - Docker giúp cô lập tài nguyên (CPU, RAM) cho từng container, giúp quản lý và tối ưu hóa tài nguyên dễ dàng hơn.
 - Docker cũng hỗ trợ các công cụ giám sát và quản lý tài nguyên, giúp bạn theo dõi và điều chỉnh khi cần thiết.

3. Bảo mật

- An toàn bảo mật:
 - Docker Official Image cho Python tuân theo các tiêu chuẩn bảo mật nghiêm ngặt, giúp giảm nguy cơ bảo mật từ các lỗ hổng phần mềm.
 - Việc đóng gói ứng dụng và các dependencies trong container giúp giảm nguy cơ từ các yếu tố bên ngoài.
- Cập nhật bảo mật:
 - Docker Official Image thường xuyên được cập nhật, bao gồm các bản vá bảo mật mới nhất.
 - Bạn cần theo dõi và cập nhật hình ảnh Docker thường xuyên để đảm bảo luôn sử dụng phiên bản an toàn nhất.

4.2 BÀN LUẬN VÀ ĐÁNH GIÁ TÍNH NĂNG TRUY CẬP VÀO LOCAL URL VÀ PUBLIC URL

1. Hiệu năng truy cập từ local link và public link

- Tốc độ phản hồi:
 - Local link: Khi truy cập qua local link, phản hồi nhanh hơn (3s) vì yêu cầu không phải đi qua nhiều mạng trung gian. Kết nối trực tiếp giữa máy tính và container Docker trên cùng một máy hoặc mạng cục bộ giúp giảm độ trễ mạng.
 - Public link: Khi truy cập qua public link, phản hồi chậm hơn (3.8s) vì yêu cầu phải đi qua mạng internet công cộng, có thể gặp phải độ trễ do nhiều yếu tố như tốc độ mạng, thời gian xử lý tại các điểm trung gian và tải mạng hiện tại.
- Nguyên nhân:
 - Độ trễ mạng: Truy cập qua internet công cộng thường có độ trễ cao hơn so với mạng cục bộ do khoảng cách và số lượng thiết bị trung gian nhiều hơn.
 - Tải mạng: Trạng thái tải của mạng internet tại thời điểm truy cập cũng ảnh hưởng đến tốc độ phản hồi.
 - Cấu trúc hạ tầng: Hạ tầng của dịch vụ Gradio hay bất kỳ dịch vụ nào cung cấp public URL có thể không tối ưu bằng việc truy cập trực tiếp qua mạng cục bộ.

2. Vấn đề với public URL thay đổi sau mỗi lần khởi động lại container

- Sự thay đổi URL:
 - Mỗi khi container được khởi động lại, Gradio cung cấp một URL public mới. Điều này gây ra sự bất tiện vì bạn phải cập nhật URL mỗi lần container khởi động lại.
 - URL thay đổi có thể gây ra gián đoạn dịch vụ và không thuận tiện cho người dùng hoặc các hệ thống cần truy cập thường xuyên.
- Nguyên nhân:
 - Dịch vụ Gradio hoặc dịch vụ tương tự cấp phát URL public mới mỗi lần ứng dụng khởi động lại như một biện pháp bảo mật và quản lý tài nguyên.
 - Cơ chế cấp phát động của các dịch vụ này khiến cho URL không cố định.

3. Đánh giá tổng thể

- Ưu điểm:
 - Local access: Truy cập qua local link có hiệu năng tốt, phản hồi nhanh, phù hợp cho môi trường phát triển và thử nghiệm cục bộ.
 - Ease of use: Sử dụng Docker Compose và Docker giúp dễ dàng triển khai và quản lý các dịch vụ liên quan đến ứng dụng của bạn.
- Nhược điểm:
 - Public access: Phản hồi từ public link chậm hơn do các yếu tố mạng như độ trễ và tải mạng.
 - URL dynamics: Việc URL public thay đổi mỗi lần container khởi động lại gây ra sự bất tiện và gián đoạn dịch vụ.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Việc triển khai ứng dụng Python sử dụng Docker Official Image cho dự án `python_app` mang lại nhiều lợi ích đáng kể, bao gồm tính nhất quán của môi trường, dễ quản lý và triển khai, cùng với sự hỗ trợ mạnh mẽ từ cộng đồng. Cấu trúc thư mục rõ ràng và hợp lý giúp dễ dàng quản lý các thành phần của dự án và đảm bảo môi trường phát triển và sản xuất giống nhau. Tuy nhiên, vẫn còn một số vấn đề cần được tối ưu hóa như hiệu quả quản lý dữ liệu lớn, bảo mật mô hình và tối ưu hóa Dockerfile để cải thiện hiệu suất và bảo mật.

Hướng phát triển:

1. Quản lý dữ liệu lớn:

- Sử dụng dịch vụ lưu trữ ngoài: Chuyển tập dữ liệu lớn như `balanced_df.csv` sang các dịch vụ lưu trữ đám mây như AWS S3, Google Cloud Storage, hoặc sử dụng cơ sở dữ liệu để quản lý dữ liệu hiệu quả hơn.

- Tối ưu hóa dữ liệu: Xem xét việc nén dữ liệu hoặc chia nhỏ tập dữ liệu để dễ quản lý và truy xuất.

2. Bảo mật mô hình:

- Bảo mật tệp mô hình: Sử dụng các công cụ mã hóa hoặc dịch vụ quản lý bí mật (như HashiCorp Vault) để bảo vệ tệp mô hình `maxent_model.pkl`.

- Quản lý truy cập: Đảm bảo rằng chỉ những thành phần cần thiết mới có quyền truy cập vào mô hình.

3. Tối ưu hóa Dockerfile:

- Giảm kích thước image: Thêm các lệnh để xóa cache và các tệp không cần thiết sau khi cài đặt thư viện, nhằm giảm kích thước của Docker image và tăng tốc độ build và deploy.

- Tối ưu hóa lệnh RUN: Sử dụng các lệnh RUN trong Dockerfile một cách hiệu quả để tối ưu hóa các lớp cache và giảm thời gian build.

4. Phân chia mã nguồn:

- Tách mã nguồn thành các module: Tách các phần xử lý văn bản, dự đoán mô hình và triển khai giao diện thành các module riêng biệt để mã nguồn dễ duy trì và mở rộng hơn.

- Sử dụng các thư viện và framework: Tận dụng các thư viện và framework để cải thiện cấu trúc và hiệu quả của mã nguồn.

5. Cải thiện tính liên tục của URL public:

- Reverse Proxy: Sử dụng một reverse proxy (như Nginx) với một domain cố định để chuyển tiếp các yêu cầu đến container, giúp duy trì URL cố định ngay cả khi container khởi động lại.

- Persistent Public URL: Cân nhắc sử dụng các dịch vụ cung cấp URL public cố định cho ứng dụng của bạn hoặc tự triển khai trên một máy chủ có IP tĩnh.

6. Automate URL update:

- Script automation: Viết script để tự động cập nhật URL mới sau khi container khởi động lại và thông báo cho người dùng hoặc hệ thống liên quan.

7. Tích hợp CI/CD:

- Tự động hóa quy trình: Thiết lập các pipeline CI/CD để tự động hóa quá trình build, test và triển khai, đảm bảo chất lượng và tính nhất quán của ứng dụng qua các giai đoạn phát triển.

- Kiểm thử và triển khai liên tục: Sử dụng các công cụ CI/CD như Jenkins, GitHub Actions, hoặc GitLab CI để cải thiện hiệu quả và tốc độ triển khai ứng dụng.

8. Khả năng mở rộng:

- Kubernetes: Sử dụng Kubernetes để quản lý các container, cung cấp các tính năng như cân bằng tải, tự động mở rộng và dịch vụ khám phá để cải thiện hiệu năng và quản lý dịch vụ tốt hơn.

TÀI LIỆU THAM KHẢO

- Buyya, R. (2009). A 2. Survey of High-Performance Computing: Architectures, Technologies, and Applications. *Journal of Parallel and Distributed Computing*, 70(3–4), 231–253.
- Containerize a Python application.* (2024, February 9). Docker Documentation. <https://docs.docker.com/language/python/containerize/>
- Guides.* (2024, May 22). Docker Documentation. <https://docs.docker.com/guides/>
- Home - welcome to ISC high performance 2024.* (n.d.). Isc-hpc.com. Retrieved May 29, 2024, from <https://www.isc-hpc.com/>
- Loeber, P. (n.d.). *dev-environment/README.md at main · patrickloeber/python-docker-tutorial.*
- Nguyen, T. (2023, March 31). *Hướng dẫn cách triển khai và debug code Python trên Docker.* Blog | Got It Vietnam; Got It Vietnam. <https://vn.got-it.ai/blog/huong-dan-cach-trien-khai-va-debug-code-python-tren-docker>
- Sterling, T., & Larson, J. F. (2020). *High Performance Computing: Architecture and Algorithms.* Cambridge University Press.