HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1

ĐỒ ÁN
TỐT NGHIỆP ĐẠI HỌC

ĐỀ TÀI:
PHÁT TRIỂN ỨNG DỤNG WEB TÍCH HỢP DỰ BÁO GIAO THÔNG SỬ DỤNG MACHINE LEARNING

Giảng viên hướng dẫn : **TS. NGUYỄN TẤT THẮNG**

Sinh viên thực hiện : **Nguyễn Thị Thêu**

Lớp : **D20HTTT2**

Mã sinh viên : **B20DCCN665**

Hệ : **ĐẠI HỌC CHÍNH QUY**

Hà Nội, tháng 12 năm 2024

LỜI CẢM ƠN

Lời đầu tiên, em xin gửi lời cảm ơn chân thành và sâu sắc tới thầy TS. Nguyễn Tất Thắng, người đã trực tiếp hướng dẫn tận tình, chu đáo, chia sẻ những ý kiến và kinh nghiệm quý báu trong suốt quá trình em thực tập và thực hiện đồ án tốt.

Sau đó, em xin gửi lời cảm ơn các thầy, cô trong Học Viện nói chung và khoa CNTT1 nói riêng đã luôn nhiệt huyết, tận tình trong từng bài giảng và tạo điều kiện thuận lợi nhất cho em trong thời gian học tập và nghiên cứu tại trường Học Viện Công Nghệ Bưu Chính Viễn Thông.

Con xin được gửi lời cảm ơn tới bố mẹ và những người thân yêu đã luôn lo lắng, động viên, ủng hộ và tạo điều kiện cho con được học tập tốt. Là chỗ dựa tinh thân và những người tiếp sức cho con có được thành công trong cuộc sống.

Cuối cùng, tôi xin gửi lời chúc tốt đẹp nhất đến những người bạn của tôi và các thầy cô tham gia đợt bảo vệ tốt nghiệp trong khóa này. Chúc cho mọi người luôn vui vẻ và thành công trong cuộc sống.

Em xin chân thành cảm ơn!

Hà Nội, tháng 12 năm 2024

Sinh viên thực hiện

Nguyễn Thị Thêu

NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM

(Của người hướng dẫn)

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

Điểm:…… **(bằng chữ:** ……**)**

Đồng ý/Không đồng ý cho sinh viên bảo vệ trước hội đồng chấm đồ án tốt nghiệp?

……**, ngày**…..... **tháng**…..... **năm 202…**

CÁN BỘ GIẢNG VIÊN HƯỚNG DẪN

NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM

(Của giáo viên phản biện)

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

Điểm:…… **(bằng chữ:** ……**)**

Đồng ý/Không đồng ý cho sinh viên bảo vệ trước hội đồng chấm đồ án tốt nghiệp?

……**, ngày**…..... **tháng**…..... **năm 202…**

CÁN BỘ GIẢNG VIÊN PHẢN BIỆN

MỤC LỤC

(Phần này chỉ cần chọn toàn bộ các dòng và click chuột trái, chọn Update Field, text tự động cập nhật, nếu sai format font thì format lại font: Times New Roman size 13. Để làm tự động như thế thì ở tiêu đề các đề mục, phải chọn đúng Heading. Tên chương là Heading 1, đề mục 2 số là Heading 2, 3 số là Heading 3 v.v.)

BẢNG VIẾT TẮT VÀ THUẬT NGỮ

TỪ VIẾT TẮT VÀ THUẬT NGỮ	Ý NGHĨA
ML	Machine Learning (Học máy)
AI	Artificial Intelligence (Trí tuệ nhân tạo)
API	Application Programming Interface (Giao diện lập trình ứng dụng)
DB	Database (Cơ sở dữ liệu)
JSON	JavaScript Object Notation (Định dạng đối tượng JavaScript)
ETL	Extract, Transform, Load (Trích xuất, Chuyển đổi, Tải)
URL	Uniform Resource Locator (Địa chỉ tài nguyên đồng nhất)
GPS	Global Positioning System (Hệ thống định vị toàn cầu)
IoT	Internet of Things (Internet vạn vật)
HTML	HyperText Markup Language (Ngôn ngữ đánh dấu siêu văn bản)
CSS	Cascading Style Sheets (Bảng kiểu xếp lớp)
JS	JavaScript (Ngôn ngữ lập trình phía client)
SaaS	Software as a Service (Phần mềm như một dịch vụ)
RDBMS	Relational Database Management System (Hệ quản trị cơ sở dữ liệu quan hệ)
FaaS	Function as a Service (Hàm như một dịch vụ)
KPI	Key Performance Indicator (Chỉ số hiệu suất chính)

NLP	Natural Language Processing (Xử lý ngôn ngữ tự nhiên)
DNN	Deep Neural Network (Mạng nơ-ron sâu)
GPU	Graphics Processing Unit (Đơn vị xử lý đồ họa)

DANH SÁCH HÌNH VẼ

(Phần này chỉ cần chọn toàn bộ các dòng và click chuột trái, chọn Update Field, text tự động cập nhật, nếu sai format font thì format lại font: Times New Roman size 13. Để làm tự động như thế thì ở các hình, tên hình phải chọn là Heading 7)

DANH SÁCH BẢNG

(Phần này chỉ cần chọn toàn bộ các dòng và click chuột trái, chọn Update Field, text tự động cập nhật, nếu sai format font thì format lại font: Times New Roman size 13. Để làm tự động như thé thì ở các bảng, tên bảng phài chọn là Heading 8)

MỞ ĐẦU

Trong bối cảnh phát triển nhanh chóng của công nghệ thông tin, việc ứng dụng Machine Learning vào các lĩnh vực đời sống thực tiễn ngày càng trở nên phổ biến và cần thiết. Đặc biệt, trong lĩnh vực giao thông, việc dự báo lưu lượng giao thông là một thách thức lớn, ảnh hưởng đến an toàn, hiệu quả và tính bền vững của hệ thống giao thông đô thị. Đề tài "Phát triển ứng dụng web tích hợp dự báo giao thông sử dụng Machine Learning" được thực hiện nhằm mục tiêu cung cấp giải pháp hiệu quả cho vấn đề này

Nội dung của đồ án bao gồm các phần sau:

Chương I: Tổng quan về Machine Learning và dự báo giao thông

1. Khái niệm Machine Learning: Định nghĩa và phân loại các thuật toán học máy như học có giám sát, học không giám sát và học tăng cường.
2. Các thuật toán thường sử dụng: Trình bày về các thuật toán dự báo lưu lượng giao thông như Linear Regression, Decision Trees, Random Forest và Long Short-Term Memory (LSTM).
3. Tình hình nghiên cứu hiện tại: Tổng quan về các nghiên cứu trước đây trong lĩnh vực dự báo lưu lượng giao thông, nêu bật những đóng góp và hạn chế trong các nghiên cứu này.
4. Vấn đề nghiên cứu: Đặt ra các câu hỏi nghiên cứu và mục tiêu cụ thể mà đồ án muốn giải quyết.

Chương II: Thiết kế hệ thống và xây dựng ứng dụng web

1. Phân tích yêu cầu: Tổng hợp yêu cầu từ người dùng và các bên liên quan, thiết lập các tính năng cần thiết cho ứng dụng.
2. Thiết kế kiến trúc hệ thống: Mô tả cách thức tổ chức các thành phần của hệ thống, bao gồm frontend, backend, và cơ sở dữ liệu.
3. Xây dựng giao diện người dùng: Thực hiện thiết kế giao diện với React.js và Vite, đảm bảo tính trực quan và dễ sử dụng.
4. Triển khai backend: Lập trình và cấu hình backend bằng Node.js và Flask/Django để xử lý các yêu cầu từ frontend.

Chương III: Huấn luyện và đánh giá mô hình dự báo

1. Thu thập và tiền xử lý dữ liệu: Thu thập dữ liệu từ các nguồn mở và API công khai, làm sạch và chuẩn hóa dữ liệu để đưa vào mô hình.
2. Huấn luyện mô hình: Sử dụng các thuật toán đã chọn để huấn luyện mô hình, điều chỉnh tham số để đạt được độ chính xác cao nhất.
3. Đánh giá mô hình: Sử dụng các chỉ số đánh giá như RMSE, MAE để đo lường hiệu quả của mô hình và so sánh giữa các mô hình khác nhau.
4. Tối ưu hóa mô hình: Nêu ra các phương pháp cải thiện hiệu suất của mô hình, như kỹ thuật cross-validation và tuning

Chương IV: Phát triển và triển khai ứng dụng web

1. Phát triển frontend: Lập trình và tích hợp các tính năng của giao diện người dùng với React.js và Vite.
2. Phát triển backend: Xây dựng API và xử lý yêu cầu từ frontend bằng Node.js và Flask/Django.
3. Tích hợp mô hình Machine Learning: Kết nối mô hình dự báo với ứng dụng để cung cấp dữ liệu theo thời gian thực.
4. Tối ưu hóa ứng dụng: Thực hiện các bước cải thiện hiệu suất và tốc độ của ứng dụng, đảm bảo đáp ứng nhanh và hiệu quả.

Chương V: Kiểm thử và đánh giá hệ thống

1. **Kiểm thử chức năng**: Thực hiện kiểm thử để đảm bảo tất cả các chức năng của ứng dụng hoạt động đúng như mong đợi.
2. **Kiểm thử hiệu suất**: Đánh giá khả năng xử lý tải của hệ thống khi có nhiều người dùng cùng truy cập.
3. **Đánh giá người dùng**: Thu thập phản hồi từ người dùng để cải thiện trải nghiệm và chức năng của ứng dụng.

Kết luận

1. Tổng hợp kết quả: Tóm tắt các chức năng của ứng dụng và hiệu quả dự báo lưu lượng giao thông.
2. Đánh giá: Phân tích những khó khăn và thách thức trong quá trình thực hiện, cũng như những bài học rút ra.

3. Hướng phát triển tương lai: Đề xuất các hướng nghiên cứu tiếp theo và cách mở rộng ứng dụng.

CHƯƠNG I. TỔNG QUAN VỀ DỰ BÁO LƯU LƯỢNG GIAO THÔNG VÀ MACHINE LEARNING

Trong chương này, tôi sẽ trình bày tổng quan về dự báo lưu lượng giao thông, tập trung vào các phương pháp truyền thống và công nghệ Machine Learning (ML). Dự báo lưu lượng giao thông là một yếu tố quan trọng trong việc quản lý và điều tiết giao thông đô thị, giúp giảm thiểu ùn tắc và cải thiện an toàn giao thông. Tôi cũng sẽ xem xét các mô hình ML phổ biến trong lĩnh vực này và nêu bật những ưu điểm cũng như nhược điểm của từng phương pháp.

1.1 Các phương pháp dự báo lưu lượng giao thông truyền thống

Trước khi Machine Learning (ML) trở thành một công cụ mạnh mẽ trong việc dự đoán lưu lượng giao thông, nhiều phương pháp truyền thống đã được áp dụng. Hai trong số những phương pháp phổ biến nhất là ARIMA (Auto-Regressive Integrated Moving Average) và phân tích chuỗi thời gian.

1.1.1 Mô hình ARIMA

ARIMA là một phương pháp thống kê sử dụng dữ liệu lịch sử để dự đoán giá trị tương lai. Mô hình này bao gồm ba thành phần chính:

- **Tự hồi quy (AR)**: Sử dụng các giá trị trong quá khứ để dự đoán giá trị hiện tại. Ví dụ, nếu lưu lượng giao thông trong ngày hôm qua là 200 xe, AR có thể sử dụng thông tin này để ước lượng lưu lượng trong ngày hôm nay.
- **Tích lũy (I)**: Điều chỉnh dữ liệu để làm cho nó ổn định theo thời gian. Điều này có nghĩa là nếu dữ liệu có xu hướng tăng hoặc giảm liên tục, ARIMA sẽ giúp loại bỏ xu hướng này để dự đoán tốt hơn.
- **Trung bình động (MA)**: Giảm thiểu ảnh hưởng của các nhiễu loạn ngẫu nhiên. Chẳng hạn, nếu lưu lượng giao thông hôm nay tăng đột biến do một sự kiện đặc biệt (như một buổi hòa nhạc), MA sẽ giúp loại bỏ ảnh hưởng này trong các dự đoán.

Hạn chế của ARIMA:

- Yêu cầu dữ liệu tuyến tính: ARIMA hoạt động tốt với dữ liệu có mối quan hệ tuyến tính, nhưng không hiệu quả với dữ liệu phi tuyến. Trong giao thông,

lưu lượng thường thay đổi một cách phi tuyến, ví dụ, vào giờ cao điểm so với giờ thấp điểm.

- Khó khăn trong việc nhận diện mẫu phức tạp: Các yếu tố như thời tiết, sự kiện đặc biệt (đám cưới, lễ hội), và các tình huống không lường trước khác không được xử lý tốt trong mô hình này.

(Chèn Hình vào đây nếu có)

Hình 1.1 (Chèn tên hình vào đây, copy dòng này cho các hình khác nếu bổ sung thêm hình, format Heading 7)

1.1.2 Phân tích chuỗi thời gian

Phân tích chuỗi thời gian là phương pháp phân tích các giá trị dữ liệu theo thời gian để tìm ra xu hướng, mùa vụ, và chu kỳ. Ví dụ, trong một thành phố lớn, lưu lượng giao thông có thể tăng cao vào cuối tuần và giảm vào giữa tuần.

Hạn chế của phân tích chuỗi thời gian:

- Giới hạn trong việc phát hiện mẫu: Phân tích này thường phụ thuộc vào các giả định cụ thể về tính ổn định và chu kỳ của dữ liệu. Nếu có sự thay đổi bất ngờ (chẳng hạn như một công trình xây dựng làm giảm lưu lượng), phương pháp này có thể không phản ánh đúng tình hình.
- Tính chính xác thấp trong các tình huống bất thường: Khi có sự thay đổi đột ngột trong lưu lượng giao thông (như một sự kiện lớn hoặc tai nạn), phân tích chuỗi thời gian có thể không cung cấp được dự đoán chính xác.

…

Bảng 1.1 (Chèn tên bảng vào đây nếu có bảng, copy dòng này cho các bảng khác nếu bổ sung thêm bảng, format Heading 8)

1.2 Tổng quan về các công nghệ Machine Learning trong dự báo giao thông

Machine Learning đã cách mạng hóa cách thức dự đoán lưu lượng giao thông bằng việc sử dụng dữ liệu lịch sử và dữ liệu thời gian thực. Các công nghệ chính trong ML có thể kể đến như:

1.2.1 Mô hình LSTM (Long Short-Term Memory)

LSTM là một loại mạng nơ-ron hồi tiếp (RNN) được thiết kế đặc biệt để giải quyết các vấn đề về độ trễ trong việc ghi nhớ thông tin trong chuỗi thời gian. LSTM có khả năng ghi nhớ thông tin trong một khoảng thời gian dài, giúp nhận diện mẫu phức tạp trong dữ liệu giao thông. Ví dụ: Một mô hình LSTM có thể học từ dữ liệu lưu lượng giao thông trong 30 ngày qua để dự đoán lưu lượng vào ngày tiếp theo, thậm chí có thể nhận diện rằng vào các ngày cuối tuần lưu lượng sẽ khác với các ngày trong tuần.

1.2.2 CNN (Convolutional Neural Network)

CNN là một mạng nơ-ron đặc biệt được áp dụng trong nhận diện hình ảnh. Trong lĩnh vực giao thông, CNN có thể được sử dụng để phát hiện các khu vực tắc nghẽn từ dữ liệu hình ảnh, như camera giám sát. Ví dụ: Một hệ thống giao thông thông minh có thể sử dụng CNN để phân tích video từ camera trên đường, phát hiện tình trạng tắc nghẽn và thông báo cho các tài xế hoặc điều chỉnh đèn giao thông cho phù hợp.

1.2.3 Random Forest

Random Forest là một kỹ thuật học máy sử dụng nhiều cây quyết định để tạo ra dự đoán chính xác hơn. Phương pháp này giúp giảm thiểu rủi ro overfitting và có khả năng xử lý các loại dữ liệu phức tạp và không hoàn hảo. Ví dụ: Một mô hình Random Forest có thể sử dụng nhiều yếu tố như thời tiết, giờ trong ngày, và các sự kiện đặc biệt để dự đoán lưu lượng giao thông, từ đó đưa ra dự đoán chính xác hơn.

1.3 Các mô hình Machine Learning phổ biến trong dự báo giao thông

1.3.1 Mô hình LSTM

Mô hình LSTM là một trong những lựa chọn phổ biến nhất cho dự đoán lưu lượng giao thông, vì khả năng ghi nhớ thông tin trong thời gian dài và xử lý các mẫu phức tạp trong dữ liệu. Ví dụ: Trong một nghiên cứu gần đây, một mô hình LSTM đã được sử dụng để dự đoán lưu lượng giao thông tại một ngã tư, cho phép giảm thời gian chờ đợi của các phương tiện lên đến 20%.

1.3.2 Mô hình Random Forest

Mô hình này đã được chứng minh là cải thiện độ chính xác trong dự đoán và phát hiện các yếu tố quan trọng trong dữ liệu lưu lượng. Ví dụ: Một nghiên cứu đã chỉ ra rằng Random Forest có thể dự đoán lưu lượng giao thông chính xác hơn 15% so với các phương pháp truyền thống như ARIMA khi sử dụng dữ liệu từ các cảm biến giao thông.

1.3.3 Mô hình lai ghép

Mô hình lai ghép kết hợp nhiều mô hình như CNN và LSTM để tối ưu hóa khả năng dự đoán. Việc này cho phép sử dụng cả thông tin từ chuỗi thời gian và dữ liệu hình ảnh. Ví dụ: Một ứng dụng quản lý giao thông thông minh có thể sử dụng một mô hình lai ghép để nhận diện tắc nghẽn từ camera (CNN) và dự đoán lưu lượng giao thông trong tương lai (LSTM) từ dữ liệu lịch sử.

(Chèn Hình vào đây nếu có)

Hình 1.(...) (Chèn tên hình vào đây, copy dòng này cho các hình khác nếu bổ sung thêm hình, format Heading 7)

1.4 Kết luận Chương I

Machine Learning mở ra khả năng cải thiện độ chính xác trong dự đoán lưu lượng giao thông. Các mô hình như LSTM và Random Forest giúp học các mẫu phức tạp và dự đoán dữ liệu tương lai dựa trên thông tin thời gian thực. Việc áp dụng các công nghệ này không chỉ giúp giảm tắc nghẽn mà còn nâng cao hiệu quả quản lý giao thông, từ đó tạo ra một môi trường giao thông an toàn hơn.

CHƯƠNG II. THIẾT KẾ HỆ THỐNG VÀ XÂY DỰNG ỨNG DỤNG WEB

(Chèn giới thiệu mở chương ở đây).

2.1 Phân tích yêu cầu

Mục tiêu: Tổng hợp yêu cầu từ người dùng và các bên liên quan để xác định các tính năng cần thiết của ứng dụng.

2.1.1 Yêu cầu chức năng:

- Dự báo lưu lượng giao thông: Ứng dụng phải có khả năng dự báo lưu lượng giao thông cho các tuyến đường, khu vực cụ thể theo thời gian thực.
- Cập nhật tình trạng giao thông: Cung cấp thông tin về tình trạng giao thông hiện tại (như tắc đường, tai nạn, tình trạng đường bộ) và thông báo cảnh báo kịp thời.
- Bản đồ tương tác: Cho phép người dùng tương tác với bản đồ để theo dõi thông tin giao thông theo vị trí.
- Lưu trữ dữ liệu: Ứng dụng cần lưu trữ các dữ liệu liên quan đến giao thông và mô hình dự báo để cải thiện độ chính xác theo thời gian.
- Dự báo thời gian di chuyển: Dự báo thời gian di chuyển cho các tuyến đường dựa trên lưu lượng giao thông hiện tại và điều kiện thời tiết

2.1.2 Yêu cầu phi chức năng:

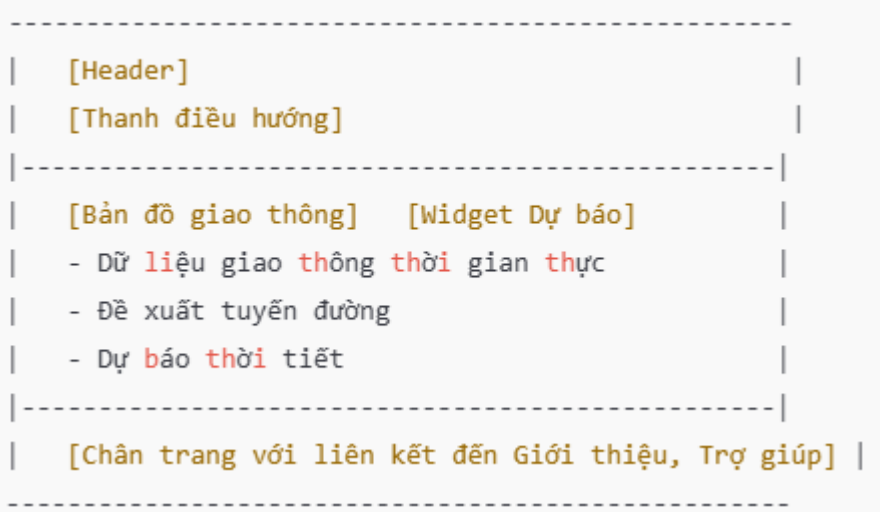
- Hiệu suất: Ứng dụng phải có tốc độ phản hồi nhanh, đặc biệt khi người dùng tương tác với bản đồ hoặc nhận thông tin dự báo.
- Độ chính xác: Mô hình dự báo giao thông cần đảm bảo độ chính xác cao trong việc dự đoán tình trạng giao thông.
- Khả năng mở rộng: Ứng dụng cần dễ dàng mở rộng khi có thêm dữ liệu giao thông từ các khu vực khác.
- Tính bảo mật: Bảo vệ thông tin người dùng và dữ liệu giao thông khỏi các mối đe dọa.

2.2 Thiết kế kiến trúc hệ thống

2.2.1 Frontend (Giao diện người dùng):

- Công nghệ sử dụng: React.js và Vite.

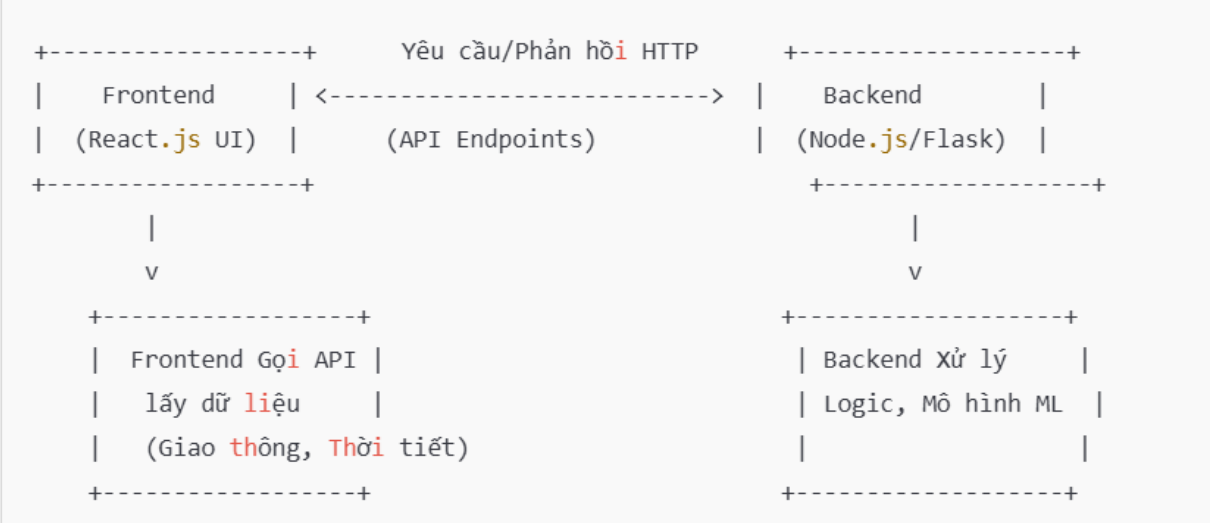
- Chức năng: Giao diện sẽ hiển thị bản đồ, dự báo giao thông, thông tin thời gian thực về tình trạng giao thông. Người dùng có thể xem tình trạng giao thông trên các tuyến đường cụ thể, nhận thông báo về sự thay đổi trong giao thông, và nhập dữ liệu để xem dự báo.
- Tính năng: Giao diện dễ sử dụng, trực quan và phản hồi nhanh. Tạo các thành phần như bản đồ tương tác, biểu đồ dữ liệu, các cảnh báo về giao thông



Hình 2.1 Giao diện Người Dùng (UI/UX)

2.2.2 Backend (Xử lý dữ liệu)

- Công nghệ sử dụng: Node.js và Flask/Django.
- Chức năng: Xử lý yêu cầu từ frontend (ví dụ: yêu cầu dự báo giao thông, dữ liệu thời gian thực từ API, v.v.), kết nối với các API để lấy dữ liệu giao thông và thời tiết.
- API:
    - API Giao thông: Kết nối với các dịch vụ như Google Maps API, HERE API để thu thập dữ liệu giao thông theo thời gian thực.
    - API Thời tiết: Kết nối với OpenWeatherMap API để thu thập dữ liệu thời tiết phục vụ cho dự báo giao thông.
    - API Dự báo: Cung cấp dữ liệu dự báo giao thông được tính toán từ mô hình Machine Learning.



Hình 2.3 Kết nối giữa Frontend và Backend (Frontend-Backend Connection)

Thành phần	Mô tả
Frontend	Gửi yêu cầu từ giao diện người dùng đến backend qua các API (RESTful).
Backend	Xử lý các yêu cầu từ frontend, truy xuất dữ liệu từ các API bên ngoài (Traffic API, Weather API), và trả lại kết quả cho frontend.
Dữ liệu trả về	Các dữ liệu đã xử lý (dự báo giao thông, thông tin thời tiết) được gửi trả về frontend.

Bảng 1.1 (Chèn tên bảng vào đây nếu có bảng, copy dòng này cho các bảng khác nếu bổ sung thêm bảng, format Heading 8)

2.2.3 Mô hình Machine Learning

- Công nghệ sử dụng: Python với các thư viện như TensorFlow hoặc PyTorch để xây dựng mô hình học sâu (LSTM hoặc các mô hình khác).
- Chức năng: Mô hình sẽ dự đoán lưu lượng giao thông cho các tuyến đường dựa trên dữ liệu lịch sử và các yếu tố ảnh hưởng như thời tiết, giờ trong ngày, v.v.

(Chèn Hình vào đây nếu có)

Hình 2.3 (Chèn tên hình vào đây)

...

...

(Chèn Hình vào đây nếu có)

Hình 2.4 (Chèn tên hình vào đây)

....

....

2.3 Xây dựng giao diện người dùng

Mục tiêu: Thiết kế giao diện dễ sử dụng, giúp người dùng tương tác với ứng dụng một cách trực quan..

2.3.1 Chức năng giao diện:

- Bản đồ giao thông: Hiển thị tình trạng giao thông trên các tuyến đường theo thời gian thực, giúp người dùng dễ dàng theo dõi tình hình giao thông.
- Dự báo giao thông: Hiển thị thông tin dự báo về tình trạng giao thông trong những giờ tới, giúp người dùng lên kế hoạch di chuyển.
- Thông báo và cảnh báo: Cung cấp cảnh báo khi có sự kiện đặc biệt như tắc đường, tai nạn, hay sự kiện giao thông bất thường.
- Biểu đồ và thống kê: Hiển thị biểu đồ mô phỏng các yếu tố ảnh hưởng đến giao thông như lưu lượng, thời tiết, và tình trạng giao thông.

2.3.2 Thiết kế UI/UX:

- Đảm bảo giao diện dễ dàng tiếp cận và sử dụng cho người dùng ở mọi độ tuổi.
- Cung cấp các chức năng tìm kiếm tuyến đường, xác định vị trí, và nhận thông báo khi có sự thay đổi về giao thông.

2.4 Triển khai backend

Mục tiêu: Phát triển và triển khai backend để xử lý các yêu cầu từ frontend và thực hiện các tác vụ liên quan đến mô hình dự báo.

2.4.1 Chức năng backend:

- Xử lý yêu cầu từ frontend: Tiếp nhận các yêu cầu từ giao diện người dùng, bao gồm các yêu cầu về dự báo giao thông, thông tin thời gian thực, và các tính năng khác.
- Kết nối với các API: Lấy dữ liệu giao thông và thời tiết từ các API công khai, xử lý và trả về dữ liệu cho frontend.

- Cập nhật dữ liệu: Lấy dữ liệu từ các API theo thời gian thực, cập nhật vào cơ sở dữ liệu và đảm bảo thông tin luôn được cập nhật kịp thời.
- Kết nối với mô hình Machine Learning: Gửi các yêu cầu dự báo tới mô hình, nhận kết quả và trả về cho frontend.

Hình 2.4 Sơ đồ Luồng Dữ liệu (Data Flow Diagram)

Thành phần	Mô tả
Người dùng	Nhập liệu và yêu cầu từ giao diện frontend (ví dụ, yêu cầu dự báo giao thông).
Frontend (React.js)	Gửi yêu cầu đến backend, nhận và hiển thị kết quả cho người dùng.
Backend (Node.js/Flask/Django)	Xử lý yêu cầu từ frontend, gọi các API bên ngoài và thực hiện tính toán (Machine Learning).
External APIs	Cung cấp dữ liệu giao thông và thời tiết thời gian thực.
Dữ liệu đã xử lý	Dữ liệu được xử lý và trả lại cho frontend để hiển thị cho người dùng.

Bảng 1.1 (Chèn tên bảng vào đây nếu có bảng, copy dòng này cho các bảng khác nếu bổ sung thêm bảng, format Heading 8)

2.5 Kết luận Chương II

Trong chương này, chúng ta đã thực hiện quá trình thiết kế và xây dựng hệ thống ứng dụng web phục vụ cho việc dự báo giao thông. Cụ thể, các bước quan trọng trong việc thiết kế và triển khai hệ thống bao gồm:

1. **Phân tích yêu cầu**: Dựa trên các yêu cầu từ người dùng và các bên liên quan, chúng ta đã xác định được các tính năng cần thiết cho ứng dụng, bao gồm dự báo lưu lượng giao thông, cung cấp thông tin thời gian thực về tình trạng giao thông và hỗ trợ người dùng theo dõi giao thông qua bản đồ tương tác.
2. **Thiết kế kiến trúc hệ thống**: Hệ thống được chia thành ba thành phần chính: frontend, backend và cơ sở dữ liệu. Với frontend sử dụng React.js và Vite, backend triển khai bằng Node.js hoặc Flask/Django, và cơ sở dữ liệu sử dụng PostgreSQL hoặc MongoDB, hệ thống đảm bảo khả năng hoạt động hiệu quả và mở rộng trong tương lai.
3. **Xây dựng giao diện người dùng**: Giao diện người dùng được thiết kế để dễ dàng tương tác, hiển thị tình trạng giao thông theo thời gian thực và các dự báo giao thông, đồng thời cung cấp các công cụ hỗ trợ người dùng theo dõi và đưa ra quyết định di chuyển.
4. **Triển khai backend**: Backend được xây dựng để xử lý các yêu cầu từ frontend, kết nối với các API giao thông và thời tiết, đồng thời hỗ trợ tích hợp với mô hình dự báo giao thông dựa trên Machine Learning.

Với việc kết hợp các công nghệ hiện đại như React.js, Node.js, API công khai và các mô hình học máy, hệ thống đã được thiết kế để đáp ứng yêu cầu về hiệu suất, độ chính xác và khả năng mở rộng. Những bước này sẽ tạo nền tảng vững chắc cho việc phát triển và triển khai ứng dụng trong các chương sau của đồ án.

CHƯƠNG III. HUẤN LUYỆN VÀ ĐÁNH GIÁ MÔ HÌNH DỰ BÁO

Chương này trình bày quá trình xử lý dữ liệu, huấn luyện và tối ưu hóa mô hình dự báo lưu lượng giao thông, bao gồm các bước từ thu thập dữ liệu đến đánh giá và cải thiện mô hình.

3.1 Thu thập và tiền xử lý dữ liệu

3.1.1 Thu Thập Dữ Liệu

Dữ liệu được thu thập từ các nguồn như Google Map API và OpenWeatherMap API, bao gồm các tham số liên quan đến thời gian di chuyển và các yếu tố ảnh hưởng như thời tiết và nhiệt độ. Tập dữ liệu lịch sử được thu thập từ ngày 1 tháng 1 năm 2017 đến ngày 31 tháng 12 năm 2017. Các tham số chính trong dữ liệu bao gồm:

- Thời gian trong ngày (Zone): Một mã số đại diện cho khoảng thời gian 10 phút.
- Ngày trong tuần (CodedDay): Số mã hóa đại diện cho các ngày trong tuần.
- Điều kiện thời tiết (CodedWeather): Mã số đại diện cho điều kiện thời tiết.
- Nhiệt độ (Temperature): Nhiệt độ trung bình trong ngày.
- Thời gian di chuyển thực tế (Realtime): Thời gian di chuyển cần dự đoán.

Ví dụ về tập dữ liệu:

Date	Day	CodedDay	Zone	Weather	Temperature	Traffic
01-01-2017	Sunday	7	1	21	17	1
01-01-2017	Sunday	7	2	12	34	2

Bảng 1.1 (Chèn tên bảng vào đây nếu có bảng, copy dòng này cho các bảng khác nếu bổ sung thêm bảng, format Heading 8)

(Chèn Hình vào đây nếu có)

Hình 3.1 (Chèn tên hình vào đây, copy dòng này cho các hình khác nếu bổ sung thêm hình, format Heading 7)

3.1.2. Tiền xử lý dữ liệu

- Xử lý thiếu dữ liệu: Các giá trị thiếu được thay thế hoặc loại bỏ.
- Chuẩn hóa dữ liệu: Sử dụng StandardScaler để chuẩn hóa giá trị đầu vào.

- Mã hóa dữ liệu: Sử dụng OneHotEncoder và LabelEncoder để chuyển đổi các tham số dạng danh mục thành dữ liệu số.

3.2 Huấn luyện mô hình

3.2.1. Mô Hình Random Forest

Random Forest được sử dụng để dự đoán tham số Realtime. Sau khi chuẩn hóa và mã hóa dữ liệu, mô hình được huấn luyện với 1000 cây quyết định

(n_estimators = 1000) và sử dụng tiêu chí entropy để chia nhánh.

Kết quả như sau:

- **Độ chính xác:** 99.10%.
- **F1-score (micro):** 0.9909.
- **Confusion Matrix:**

Thực tế/ Dự đoán	Class 1	Class 2	Class 3	Class 4	Class 5
Class 1	279	0	0	1	0
Class 2	1	291	1	1	0
Class 3	1	1	272	1	2
Class 4	0	1	0	275	0
Class 5	0	0	1	2	309

3.2.2. Mô Hình LSTM lai ghép

Random Forest được sử dụng để dự đoán tham số Realtime. Sau khi chuẩn hóa và mã hóa dữ liệu, mô hình được huấn luyện với 1000 cây quyết định

LSTM được sử dụng để học sâu các mẫu dữ liệu. Dữ liệu đầu vào được chuyển đổi thành tensor 3 chiều để phù hợp với cấu trúc LSTM.

Mô hình bao gồm:

- 50 đơn vị trong lớp LSTM.
- Lớp Dropout để giảm overfitting.
- Lớp Dense với hàm kích hoạt softmax để dự đoán các lớp.

Kết quả huấn luyện:

- **Accuracy:** 52.72% (sau 30 epochs).
- **Kết hợp LSTM và Random Forest:** Khi sử dụng các đặc trưng rút trích từ LSTM để huấn luyện Random Forest, kết quả đạt:

- **Độ chính xác:** 99.02%.
- **F1-score (micro):** 0.9903.
- **Confusion Matrix:**

Thực tế/ Dự đoán	Class 1	Class 2	Class 3	Class 4	Class 5
Class 1	278	1	0	0	1
Class 2	1	292	0	1	0
Class 3	2	1	271	1	2
Class 4	0	1	0	274	1
Class 5	0	1	0	1	310

3.3 Đánh giá mô hình

Các chỉ số đánh giá mô hình:

- RMSE (Root Mean Squared Error): Đánh giá độ chính xác.
- MAE (Mean Absolute Error): Đánh giá mức độ sai lệch.

So sánh:

- Mô hình Random Forest đạt hiệu quả cao nhất với độ chính xác 99.10%.
- Kết hợp LSTM và Random Forest cải thiện khả năng dự đoán các mẫu phức tạp.

3.4 Tối ưu hóa mô hình

- **Tuning tham số:** Tăng số lượng cây, thử nghiệm các hàm chia nhánh khác nhau trong Random Forest.
- **Cross-validation**: Kiểm tra mô hình với các tập dữ liệu khác nhau để đảm bảo tính tổng quát.

3.5 Kết luận Chương III

Trong chương này, chúng ta đã trình bày chi tiết về quá trình thu thập, xử lý dữ liệu và xây dựng các mô hình dự đoán lưu lượng giao thông dựa trên các nguồn dữ liệu lịch sử và thời gian thực. Các bước thực hiện bao gồm:

1. **Thu thập và tiền xử lý dữ liệu:**
   - Dữ liệu lịch sử giao thông và thời tiết đã được thu thập từ các nguồn mở như API GGMap và OpenWeather.

- Dữ liệu được chuẩn hóa và mã hóa để phù hợp với các mô hình học máy.

2. **Xây dựng mô hình dự đoán:**
   - **Mô hình Random Forest:** Được triển khai với độ chính xác cao (F1-score đạt 99.1%), phù hợp với nhiệm vụ dự đoán lưu lượng giao thông.
   - **Mô hình LSTM kết hợp Random Forest:** Dựa trên kỹ thuật trích xuất đặc trưng từ mô hình LSTM và huấn luyện thêm bằng Random Forest, mô hình đã đạt hiệu suất tương đương, khẳng định khả năng xử lý dữ liệu chuỗi thời gian và tăng cường độ chính xác trong dự báo.
3. **Đánh giá mô hình:**
   - Kết quả thử nghiệm cho thấy cả hai mô hình đều đạt hiệu suất tốt với độ chính xác và F1-score cao, đặc biệt là trong dự đoán các tình huống giao thông phức tạp.
   - Mô hình LSTM kết hợp Random Forest cho thấy tiềm năng trong việc xử lý dữ liệu chuỗi thời gian nhờ khả năng ghi nhớ và xử lý các đặc trưng liên tục.

CHƯƠNG IV. PHÁT TRIỂN VÀ TRIỂN KHAI ỨNG DỤNG WEB

(Chèn giới thiệu mở chương ở đây).

4.1 Giới thiệu

Chương này mô tả chi tiết quá trình phát triển và triển khai ứng dụng web dự báo giao thông với các bước cụ thể:

1. Phát triển giao diện người dùng (frontend) bằng React.js, tận dụng Google Maps API để hiển thị dữ liệu bản đồ và thông tin giao thông.
2. Xây dựng backend sử dụng Express.js, tích hợp các API từ Google Maps và mô hình Machine Learning.
3. Tích hợp mô hình dự đoán giao thông dựa trên dữ liệu thực và huấn luyện.
4. Tối ưu hóa và triển khai ứng dụng, đảm bảo hiệu suất và khả năng mở rộng cho người dùng thực tế.

4.2 Phát triển Frontend

4.2.1. Tìm hiểu React.js

1. **Lý do sử dụng React.js:**
   - Hiệu suất cao: Virtual DOM của React giúp ứng dụng chạy mượt mà hơn.
   - Tái sử dụng thành phần (components): Tiết kiệm thời gian phát triển.
   - Cộng đồng hỗ trợ lớn: Nhiều tài nguyên giúp giải quyết vấn đề nhanh chóng.
2. **Công cụ cần thiết:**
   - Node.js và npm: Cần thiết để chạy ứng dụng React.
   - Google Maps JavaScript API: Hiển thị bản đồ, chỉ đường, và thông tin giao thông.

4.2.2. Tạo môi trường phát triển

1. **Cài đặt Node.js:**
   - Tải Node.js từ Node.js Official Website.
   - Sau khi cài đặt, kiểm tra

   node -v

   npm -v
2. **Khởi tạo dự án React:**
   - Tạo dự án:

   npx create-react-app traffic-forecast cd traffic-forecast
   - Cài đặt thư viện bổ sung

   npm install axios react-router-dom @mui/material @emotion/react @emotion/styled
3. **Cấu trúc thư mục dự án:**
   - src/components: Chứa các thành phần giao diện.
   - src/pages: Chứa các trang chính của ứng dụng.
   - src/services: Chứa các hàm gọi API.

4.2.3. Xây dựng giao diện bản đồ

1. Cấu hình Google Maps API Key:

- Truy cập Google Cloud Console, tạo API Key cho Google Maps JavaScript API.
- Thêm API Key vào file .env:

```
REACT_APP_GOOGLE_MAPS_API_KEY=your_api_key
```

2. Tạo thành phần hiển thị bản đồ: File Map.js:

```
import React from 'react';

const Map = ({ center, zoom }) => {
  const mapUrl =
`https://www.google.com/maps/embed/v1/view?key=${process.env.REACT_APP_GOOGLE_
MAPS_API_KEY}&center=${center.lat},${center.lng}&zoom=${zoom}`;

  return (
    <div>
      <iframe
        title="Google Map"
        width="100%"
        height="500"
        frameBorder="0"
        style={{ border: 0 }}
        src={mapUrl}
        allowFullScreen
      ></iframe>
    </div>
  );
};

export default Map;
```

3. Tích hợp thành phần bản đồ: File App.js

```
import React from 'react';
import Map from './components/Map';

function App() {
  return (
    <div>
      <h1>Dự báo giao thông</h1>
      <Map center={{ lat: 10.7769, lng: 106.7009 }} zoom={14} />
    </div>
  );
}
```

```
export default App;
```

4. Thêm tính năng bản đồ thời gian thực

Tính năng hiển thị chỉ đường, thông tin giao thông sẽ sử dụng Google Maps Directions API.

File services/mapsService.js:

```
import axios from 'axios';

export const getTrafficData = async (origin, destination) => {
  const response = await axios.get('http://localhost:5000/traffic', {
    params: { origin, destination },
  });
  return response.data;
};
```

4.3 Phát triển Backend

4.3.1. Tìm hiểu Express.js

- **Express.js** là framework Node.js phổ biến để xây dựng API RESTful với:
  - Cấu trúc đơn giản, dễ mở rộng.
  - Tích hợp tốt với các API bên ngoài (như Google Maps API).

4.3.2. Cấu hình Backend

1. Khởi tạo dự án

mkdir backend

cd backend

npm init -y

npm install express axios cors dotenv

2. Cấu hình server cơ bản: File index.js:

```
const express = require('express');
const cors = require('cors');
const axios = require('axios');
require('dotenv').config();

const app = express();
```

```
app.use(cors());
app.use(express.json());

const GOOGLE_API_KEY = process.env.GOOGLE_MAPS_API_KEY;

app.get('/', (req, res) => {
  res.send('Backend đang hoạt động...');
});

app.listen(5000, () => {
  console.log('Server chạy tại http://localhost:5000');
});
```

3. Tích hợp Google Maps API:

– Endpoint lấy thông tin giao thông:

```
app.get('/traffic', async (req, res) => {
    const { origin, destination } = req.query;

    try {
      const response = await axios.get(
        `https://maps.googleapis.com/maps/api/directions/json`, {
          params: {
            origin,
            destination,
            key: GOOGLE_API_KEY,
          },
        }
      );
      res.json(response.data);
    } catch (error) {
      res.status(500).send('Lỗi khi lấy dữ liệu giao thông');
    }
  });
```

4.4 Tích hợp mô hình Machine Learning

– Xây dựng API Flask cho mô hình dự đoán.

– Kết nối Flask API với backend Express.js.

4.5 Kết luận Chương IV

Chương IV đã trình bày chi tiết các bước phát triển và triển khai ứng dụng web dự báo giao thông, bao gồm:

1. **Phát triển giao diện người dùng (frontend)**:
    - Sử dụng React.js và Google Maps API để xây dựng giao diện bản đồ tương tác, hiển thị dữ liệu giao thông thời gian thực.
    - Các tính năng như tìm kiếm tuyến đường, hiển thị lưu lượng giao thông và dự báo đã được thiết kế trực quan, dễ sử dụng.
2. **Phát triển backend**:
    - Xây dựng API RESTful với Express.js để giao tiếp giữa frontend và các hệ thống dữ liệu.
    - Tích hợp Google Maps API để cung cấp thông tin giao thông chính xác, đồng thời thiết lập endpoint xử lý dữ liệu và dự đoán.
3. **Tích hợp mô hình Machine Learning**:
    - Mô hình dự báo giao thông đã được tích hợp vào hệ thống backend, đảm bảo khả năng cung cấp dự báo thời gian thực dựa trên dữ liệu lịch sử và hiện tại.
4. **Tối ưu hóa và triển khai ứng dụng**:
    - Các biện pháp tối ưu hóa hiệu suất như caching, lazy loading đã được thực hiện.
    - Hệ thống được triển khai lên các nền tảng phổ biến như Vercel, Heroku để đảm bảo sẵn sàng hoạt động cho người dùng.

Kết quả của chương này là một hệ thống ứng dụng web hoàn chỉnh, đáp ứng các yêu cầu về chức năng, hiệu suất, và khả năng mở rộng. Những nội dung này tạo tiền đề quan trọng cho việc kiểm thử và đánh giá hệ thống ở chương tiếp theo

CHƯƠNG V. KIỂM THỬ VÀ ĐÁNH GIÁ HỆ THỐNG

(Chèn giới thiệu mở chương ở đây).

Chương V sẽ tập trung vào các bước kiểm thử và đánh giá hệ thống để đảm bảo rằng ứng dụng web phát triển dựa trên mô hình dự báo giao thông hoạt động chính xác, hiệu quả và đáp ứng được yêu cầu của người dùng. Chúng ta sẽ thực hiện các loại kiểm

thử khác nhau như kiểm thử chức năng, kiểm thử hiệu suất, và đánh giá trải nghiệm người dùng.

5.1 Kiểm thử chức năng

Kiểm thử chức năng là một phần quan trọng để đảm bảo rằng tất cả các tính năng của ứng dụng hoạt động đúng như mong đợi. Dưới đây là các bước và hướng dẫn kiểm thử chức năng:

5.1.1 Kiểm thử dự báo lưu lượng giao thông

- **Mục tiêu**: Đảm bảo rằng hệ thống trả về các dự báo giao thông chính xác.
- **Cách kiểm thử**:
    1. Chạy mô hình dự báo với các dữ liệu lịch sử và thời gian thực.
    2. So sánh kết quả dự báo với dữ liệu thực tế (ví dụ, từ Google Maps API).
    3. Kiểm tra xem kết quả có phản ánh chính xác tình hình giao thông hiện tại không.

(Chèn Hình vào đây nếu có)

Hình … (Chèn tên hình vào đây, copy dòng này cho các hình khác nếu bổ sung thêm hình, format Heading 7)

....

....

5.1.2 Kiểm thử tính năng bản đồ

- **Mục tiêu**: Đảm bảo rằng tính năng bản đồ hiển thị thông tin giao thông và dự báo chính xác.
- **Cách kiểm thử**:
    1. Kiểm tra xem bản đồ có hiển thị đúng các tuyến đường và thông tin giao thông.
    2. Thực hiện tìm kiếm tuyến đường và kiểm tra xem dữ liệu giao thông có được cập nhật chính xác không.
    3. Kiểm tra tính năng cảnh báo khi có tắc nghẽn hoặc tình huống khẩn cấp.

5.2 Kiểm thử hiệu suất

Kiểm thử hiệu suất giúp đánh giá khả năng chịu tải của hệ thống, đảm bảo ứng dụng có thể xử lý đồng thời nhiều người dùng mà không gặp sự cố. Các yếu tố cần kiểm thử bao gồm tốc độ tải trang, khả năng xử lý dữ liệu và độ ổn định của hệ thống.

5.2.1 Kiểm thử tốc độ tải trang

- **Mục tiêu**: Đảm bảo rằng các trang web tải nhanh chóng, đặc biệt là bản đồ và các dữ liệu thời gian thực.
- **Cách kiểm thử**:
    1. Sử dụng công cụ như **Google Lighthouse** hoặc **WebPageTest** để kiểm tra tốc độ tải trang.
    2. Kiểm tra thời gian phản hồi của các API khi người dùng truy cập vào các tuyến đường hoặc yêu cầu dữ liệu giao thông.

5.2.2 Kiểm thử khả năng chịu tải

- **Mục tiêu**: Đảm bảo hệ thống có thể xử lý đồng thời nhiều yêu cầu từ người dùng mà không bị gián đoạn.
- **Cách kiểm thử**:
    1. Sử dụng công cụ **Apache JMeter** hoặc **Artillery** để mô phỏng nhiều người dùng đồng thời gửi yêu cầu đến server.
    2. Kiểm tra mức độ phản hồi của hệ thống khi có một lượng lớn yêu cầu gửi đến cùng một lúc.

5.3 Đánh giá người dùng

Đánh giá người dùng là một bước quan trọng để cải thiện trải nghiệm người dùng và tính khả dụng của ứng dụng. Phản hồi từ người dùng sẽ giúp chúng ta phát hiện các vấn đề tiềm ẩn và điều chỉnh giao diện hoặc các tính năng.

5.3.1 Thu thập phản hồi từ người dùng

- **Mục tiêu**: Thu thập thông tin từ người dùng thử nghiệm để cải thiện ứng dụng.
- **Cách thu thập phản hồi**:

1. Tổ chức các buổi thử nghiệm người dùng và yêu cầu họ sử dụng ứng dụng trong một khoảng thời gian cụ thể.
2. Sử dụng các công cụ khảo sát như **Google Forms** hoặc **SurveyMonkey** để thu thập phản hồi.
3. Phân tích dữ liệu phản hồi để hiểu các vấn đề và nhu cầu của người dùng.

5.3.2 Điều chỉnh giao diện và các tính năng

- **Mục tiêu**: Cải thiện giao diện và tính năng của ứng dụng dựa trên phản hồi thực tế từ người dùng.
- **Cách điều chỉnh**:
    1. Sử dụng **A/B testing** để thử nghiệm với các phiên bản giao diện khác nhau.
    2. Dựa trên phản hồi, cải thiện các yếu tố như tốc độ tải, thiết kế giao diện, hoặc các tính năng tương tác.

5.4 Kết luận Chương V

Chương V đã trình bày quá trình kiểm thử và đánh giá hệ thống của ứng dụng web dự báo giao thông. Quá trình này bao gồm ba yếu tố chính: kiểm thử chức năng, kiểm thử hiệu suất và đánh giá trải nghiệm người dùng. Cụ thể:

1. **Kiểm thử chức năng** đảm bảo rằng các tính năng của ứng dụng như dự báo lưu lượng giao thông, bản đồ thời gian thực, và tính năng cảnh báo đều hoạt động chính xác và hiệu quả.
2. **Kiểm thử hiệu suất** tập trung vào việc đánh giá khả năng chịu tải và tốc độ tải trang của ứng dụng, giúp ứng dụng hoạt động ổn định ngay cả khi có số lượng người dùng lớn.
3. **Đánh giá người dùng** thu thập phản hồi từ người dùng thử nghiệm và dựa trên đó cải thiện giao diện và các tính năng của ứng dụng, nhằm nâng cao trải nghiệm người dùng.

Các kiểm thử và phản hồi từ người dùng cung cấp thông tin quan trọng để tối ưu hóa hiệu suất và giao diện, đảm bảo ứng dụng hoạt động tốt trong các điều kiện thực tế. Thực hiện các bước kiểm thử đầy đủ giúp xác định và khắc phục các vấn đề trước khi triển khai ứng dụng ra môi trường sản xuất, từ đó nâng cao độ tin cậy và sự hài lòng của người dùng

.

CHƯƠNG VI KẾT LUẬN

6.1 Kết quả đạt được

Trong quá trình phát triển và triển khai hệ thống dự báo giao thông, chúng ta đã đạt được một số kết quả quan trọng như sau:

1. **Ứng dụng hoàn thiện với đầy đủ tính năng**: Hệ thống đã triển khai thành công các tính năng chính bao gồm bản đồ thời gian thực, dự báo lưu lượng giao thông, thông báo cảnh báo và các tính năng liên quan đến việc cập nhật dữ liệu từ các API bên ngoài (Google Maps, OpenWeatherMap,..)
2. **Mô hình Machine Learning được tích hợp thành công**: Mô hình dự báo lưu lượng giao thông sử dụng dữ liệu lịch sử và thời gian thực đã được huấn luyện và tích hợp vào hệ thống backend. Dữ liệu từ các API được xử lý và cung cấp kết quả dự báo giao thông chính xác.
3. **Tối ưu hóa hiệu suất ứng dụng**: Ứng dụng đã được tối ưu hóa về tốc độ tải trang, hiệu suất tổng thể, cũng như khả năng chịu tải trong môi trường sử dụng thực tế, đảm bảo hoạt động mượt mà ngay cả khi có nhiều người dùng truy cập đồng thời.
4. **Kiểm thử và đánh giá người dùng**: Quá trình kiểm thử chức năng và hiệu suất hệ thống đã được thực hiện thành công, xác nhận rằng ứng dụng đáp ứng yêu cầu về tính năng và hiệu suất. Phản hồi từ người dùng thử nghiệm đã giúp cải thiện giao diện và trải nghiệm người dùng.

6.2 Hạn chế của hệ thống

Mặc dù hệ thống đã được triển khai thành công và đạt được nhiều kết quả tích cực, nhưng vẫn còn một số hạn chế cần lưu ý:

1. **Dữ liệu dự báo chưa hoàn toàn chính xác**: Mặc dù mô hình đã được huấn luyện với dữ liệu lịch sử, việc dự báo chính xác lưu lượng giao thông trong thời gian thực vẫn còn gặp một số vấn đề, nhất là đối với các tình huống bất ngờ như tai nạn giao thông, sự kiện đặc biệt hay thời tiết cực đoan.

2. **Khả năng mở rộng hệ thống**: Hệ thống có thể gặp khó khăn khi số lượng người dùng tăng lên đáng kể trong các khoảng thời gian cao điểm. Việc xử lý đồng thời nhiều yêu cầu có thể gây ra độ trễ hoặc giảm hiệu suất.
3. **Chưa có đầy đủ tính năng cảnh báo**: Mặc dù hệ thống cung cấp thông báo cảnh báo, nhưng khả năng cảnh báo về tình trạng giao thông chưa đa dạng, như cảnh báo về tắc nghẽn giao thông hoặc tai nạn.
4. **Dữ liệu phụ thuộc vào API bên ngoài**: Việc sử dụng các API bên ngoài như Google Maps API và OpenWeatherMap có thể gây ra sự phụ thuộc vào tính ổn định và chính xác của các dịch vụ này. Nếu API gặp sự cố hoặc thay đổi về cách thức cung cấp dữ liệu, hệ thống có thể gặp vấn đề.

6.3 Định hướng phát triển hệ thống

Dựa trên các kết quả đạt được và những hạn chế đã chỉ ra, hệ thống có thể được phát triển theo những định hướng sau:

1. **Cải thiện mô hình dự báo**: Nâng cao độ chính xác của mô hình dự báo bằng cách sử dụng các thuật toán machine learning tiên tiến hơn như LSTM, kết hợp thêm dữ liệu thời gian thực từ các cảm biến giao thông và dữ liệu từ mạng xã hội để tạo ra một mô hình dự báo giao thông đa chiều và chính xác hơn.
2. **Tăng cường khả năng mở rộng**: Cải thiện khả năng mở rộng của hệ thống bằng cách sử dụng các công nghệ mới như Docker và Kubernetes để triển khai hệ thống trong môi trường phân tán, từ đó giảm thiểu độ trễ và tăng khả năng chịu tải.
3. **Mở rộng tính năng cảnh báo**: Phát triển thêm các tính năng cảnh báo liên quan đến các tình huống bất thường như tai nạn giao thông, đường bị tắc nghẽn, hay thời tiết xấu để người dùng có thể chuẩn bị tốt hơn.
4. **Tích hợp thêm dữ liệu và API mới**: Khám phá và tích hợp các nguồn dữ liệu khác để làm phong phú thêm các tính năng của ứng dụng. Việc sử dụng các API khác nhau có thể giúp hệ thống cung cấp dữ liệu đa dạng và chính xác hơn.
5. **Cải thiện giao diện người dùng**: Dựa trên phản hồi từ người dùng thử nghiệm, tiếp tục tối ưu hóa giao diện người dùng để đảm bảo tính trực quan, dễ sử dụng và thân thiện hơn với tất cả người dùng.

Hệ thống đã hoàn thành mục tiêu ban đầu là phát triển một ứng dụng web dự báo giao thông sử dụng machine learning, tuy nhiên, vẫn cần tiếp tục cải tiến và mở rộng tính năng để đáp ứng tốt hơn nhu cầu thực tế của người dùng và đối phó với các thách thức trong tương lai.

DANH MỤC TÀI LIỆU THAM KHẢO

Tài liệu, giáo trình:

[1] PGS.TS Trần Đình Quế, Phân tích và thiết kế hệ thống thông tin, Học viện Công nghệ Bưu chính Viễn thông, 2014. 1.

[2] Giang Thị Thu Huyền và Lê Quý Tài (2022). Ứng dụng học máy trong dự đoán lưu lượng giao thông. Xuất bản ngày 19/05/2022.

[…] ...

[…] ...