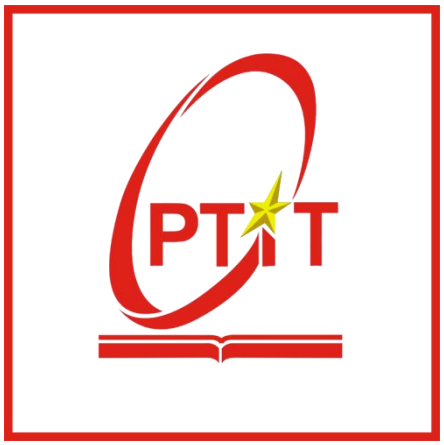
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH



BÁO CÁO CUỐI KỲ

IOT VÀ ỨNG DỤNG

ĐỀ TÀI: NGHIÊN CỨU MACHINE LEARNING MODEL PIPELINE

Giảng viên hướng dẫn : ĐÀM MINH LỊNH

Sinh viên thực hiện	Mã sinh viên	Lớp
Nguyễn Phan Nhựt Trường	N20DCCN082	D20CQCNPM01-N
Bùi Tuấn Anh	N20DCCN002	D20CQCNPM01-N

Tp. Hồ Chí Minh - 01/2024

Lời nói đầu

Kính chào thầy,

Nhóm chúng em rất hân hạnh được ở đây để chia sẻ với thầy về đề tài mà nhóm đã lựa chọn cho dự án nghiên cứu của mình. Đề tài của nhóm xoay quanh việc xây dựng một mô hình dự đoán tỷ lệ đột quỵ, và nhóm muốn giới thiệu một số lý do quan trọng khiến cho đề tài này trở nên đặc biệt và đáng quan tâm.

Một trong những nguyên nhân chính khiến nhóm quyết định nghiên cứu về dự đoán tỷ lệ đột quỵ là sự nghiên cứu này có thể mang lại giá trị lớn trong lĩnh vực y tế. Đột quỵ, hay còn được biết đến là tai biến mạch máu não, là một trong những nguyên nhân hàng đầu gây tử vong và tác động lớn đến chất lượng cuộc sống của những người mắc bệnh. Việc có thể dự đoán và đưa ra các biện pháp phòng ngừa hiệu quả sẽ giúp giảm nguy cơ và cải thiện sức khỏe cộng đồng.

Hơn nữa, dữ liệu y tế hiện nay đang trở nên ngày càng phong phú và dồi dào. Qua việc sử dụng các phương pháp và mô hình học máy, chúng ta có cơ hội để tận dụng những thông tin này để xây dựng những mô hình dự đoán chính xác. Điều này không chỉ hỗ trợ các chuyên gia y tế trong việc đưa ra các quyết định chuẩn đoán và điều trị mà còn có thể làm tăng khả năng tiên đoán và ngăn chặn các trường hợp đột quỵ trong tương lai.

Cuối cùng, đây cũng là một cơ hội để áp dụng kiến thức trong lĩnh vực khoa học dữ liệu và học máy vào các vấn đề thực tế. Sự kết hợp giữa lý thuyết và thực tiễn trong việc xây dựng mô hình có thể có ảnh hưởng tích cực đến cả sự phát triển cá nhân và sự đóng góp vào cộng đồng y tế.

Như vậy, với những lý do trên, nhóm tin rằng đề tài này không chỉ mang lại kiến thức mới mẻ mà còn đóng góp tích cực vào lĩnh vực y tế và nghiên cứu khoa học dữ liệu.

Để hoàn thành đề tài, em muốn bày tỏ lòng biết ơn sâu sắc đến Thầy Đàm Minh Lịnh - người đã trực tiếp hướng dẫn và hỗ trợ em trong suốt quá trình làm. Thầy đã cung cấp cho chúng em những ý kiến quý báu và chỉ dẫn chính xác để giúp nhóm hoàn thiện tốt đề tài.

Cảm ơn thầy đã đọc và hãy cùng nhau khám phá hơn về hành trình này.

Mục lục

I. Random Forest

1. Cây quyết định

Thuật toán của cây quyết định được mô tả như sau:

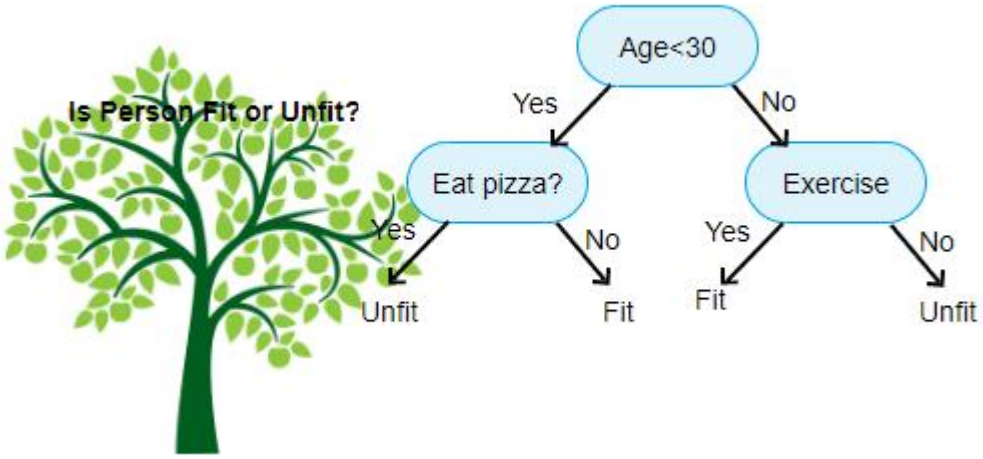
1. Chọn thuộc tính tốt nhất để làm nút gốc của cây. Thuộc tính tốt nhất là thuộc tính có khả năng phân loại tốt nhất dữ liệu huấn luyện

2. Chia tập dữ liệu huyến luyện thành các tập con dựa trên giá trị của thuộc tính được chọn. Mỗi tập con tương ứng với một giá trị của thuộc tính.

3. Đối với mỗi tập con, tiếp tục chọn thuộc tính tốt nhất để làm nút con (child node) tiếp theo. Lặp lại quá trình chia tập dữ liệu cho đến khi đạt được các điều kiện dừng, ví dụ như khi không còn thuộc tính để chọn hoặc khi các tập dữ liệu thuộc cùng một lớp.

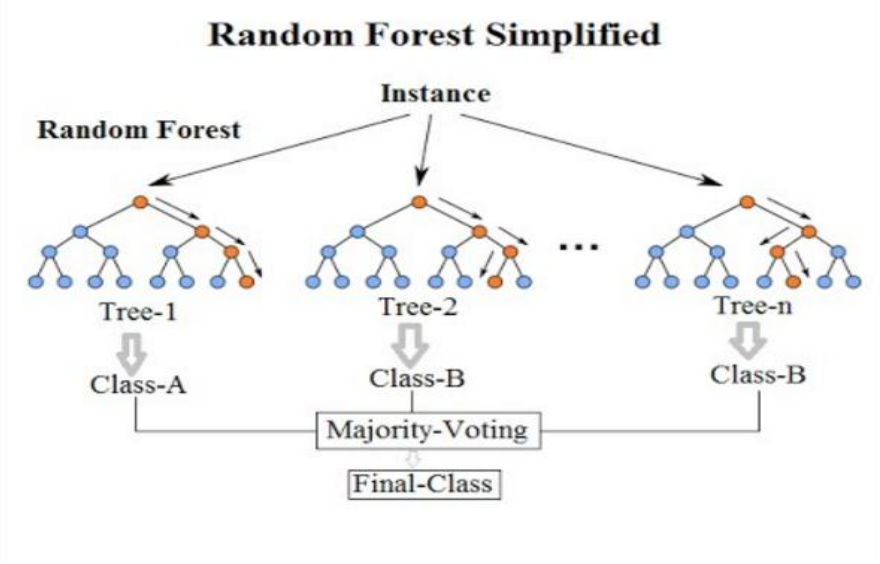
4. Gán nhãn cho các nút lá (leaf nodes) dựa trên phân phối lớp của các điểm dữ liệu trong tập huấn luyện. Một nút lá có thể đại diện cho một lớp duy nhất hoặc một giá trị dự đoán.

5. Cây quyết định đã được xây dựng và có thể được sử dụng để dự đoán nhãn cho các điểm dữ liệu mới bằng cách di chuyển từ gốc đến nút lá tương ứng với giá trị của các thuộc tính.



2. Thuật toán Random Forest

Thuật toán Random Forest là một thuật toán học có giám sát (supervised learning) được sử dụng cho cả phân lớp và hồi quy. Thuật toán này được xây dựng dựa trên việc kết hợp nhiều cây quyết định (decision trees) để tạo ra một mô hình dự đoán chính xác và ổn định hơn.



Dưới đây là một số điểm quan trọng về Random Forest:

- Tạo ra Nhiều Cây Quyết Định:

Mỗi cây quyết định được huấn luyện trên một tập dữ liệu con được chọn ngẫu nhiên từ tập dữ liệu huấn luyện.

Mỗi cây đưa ra một dự đoán riêng và sau đó kết hợp thông qua phương pháp đa số (voting) để đưa ra dự đoán cuối cùng.

- Lựa Chọn Ngẫu Nhiên:

Đối với mỗi cây quyết định, một số lượng ngẫu nhiên các đặc trưng (features) được chọn để xây dựng cây.

Điều này giúp ngăn chặn sự chệch (bias) của mỗi cây và làm cho mô hình tổng thể linh hoạt hơn.

- Tính Ổn Định và Chống Quá Mức (Overfitting):

Sự kết hợp của nhiều cây giúp giảm nguy cơ quá mức và làm cho mô hình chung ổn định hơn.

Mô hình này thường đưa ra dự đoán tốt trên dữ liệu huấn luyện và dữ liệu mới.

- Xác Định Độ Quan Trọng của Đặc Trưng:

Random Forest có thể cung cấp thông tin về độ quan trọng của mỗi đặc trưng trong việc đưa ra dự đoán.

- Ứng Dụng Rộng Rãi:

Random Forest được sử dụng trong nhiều ứng dụng như phân loại, dự đoán và giải quyết vấn đề hồi quy.

Thuật toán Random Forest được chú ý và sử dụng rộng rãi do khả năng linh hoạt và khả năng xử lý nhiễu tốt. Tuy nhiên, nhược điểm của nó bao gồm việc tăng chi phí tính toán và khó khăn trong việc giải thích mô hình.

Các bước xây dựng thuật toán Random Forests:

- Chọn ngẫu nhiên một tập con của dữ liệu huấn luyện.

- Xây dựng một cây quyết định trên tập con này.

- Lặp lại các bước 1 và 2 k lần (k là số lượng cây quyết định được chọn) để xây dựng nhiều cây quyết định khác nhau.

- Đưa ra dự đoán cho một điểm dữ liệu mới bằng cách đưa nó qua tất cả các cây quyết định và tính toán số lượng phiếu bầu cho mỗi lớp.

- Chọn lớp có số phiếu bầu cao nhất là kết quả cuối cùng.

Ưu điểm của thuật toán Random Forest: Random forests được coi là một phương pháp chính xác và mạnh mẽ vì số cây quyết định tham gia vào quá trình này

- Khả năng xử lý các tập dữ liệu lớn.

- Khả năng xử lý các tập dữ liệu có nhiều biến.

- Khả năng xử lý các tập dữ liệu có giá trị thiếu.

- Giải quyết khá tốt (nhưng không hoàn toàn) vấn đề overfitting.

- Khả năng xác định mức độ quan trọng của các biến trong mô hình.

- Khả năng xác định mức độ chính xác của mô hình.

- Có thể được sử dụng trong cả hai vấn đề phân loại và hồi quy.

Nhược điểm của thuật toán Random Forest: Random forests chậm tạo dự đoán bởi vì nó có nhiều cây quyết định.

- Có thể bị overfitting nếu số cây quá lớn.

- Không thể giải thích kết quả dự đoán một cách chi tiết như các thuật toán khác như cây quyết định.

Thuật toán Random Forest được sử dụng trong nhiều lĩnh vực khác nhau như:

-Phân loại hình ảnh

-Dự đoán giá cổ phiếu.

-Dự đoán giá nhà đất.

-Phân loại bệnh tim mạch

-Phân tích tín dụng

-Dự đoán giá nhà

-Dự đoán khả năng khách hàng sẽ mua sản phẩm.

-Dự đoán khả năng khách hàng sẽ hủy đơn hàng.

-Dự đoán khả năng khách hàng sẽ trả nợ.

-Dự đoán khả năng khách hàng sẽ chuyển đổi.

-Nhiều lĩnh vực khác…

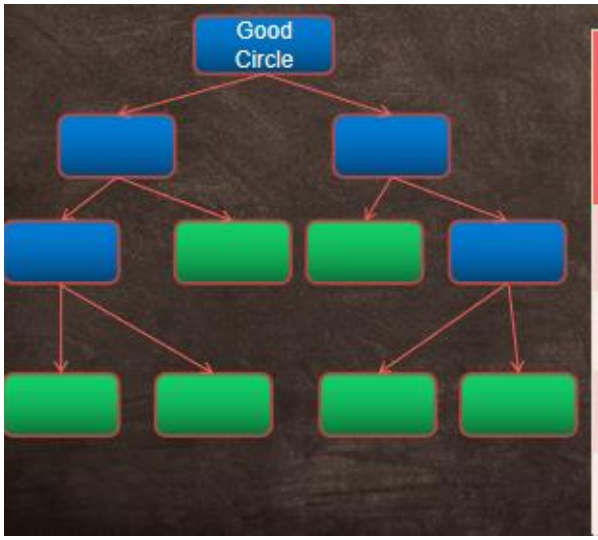
Nó được sử dụng trong các bài toán phân loại và hồi quy với tập dữ liệu lớn và có nhiều biến. Nó cũng được sử dụng khi cần xác định mức độ quan trọng của các biến trong mô hình.

VÍ DỤ VỀ RANDOM FOREST:

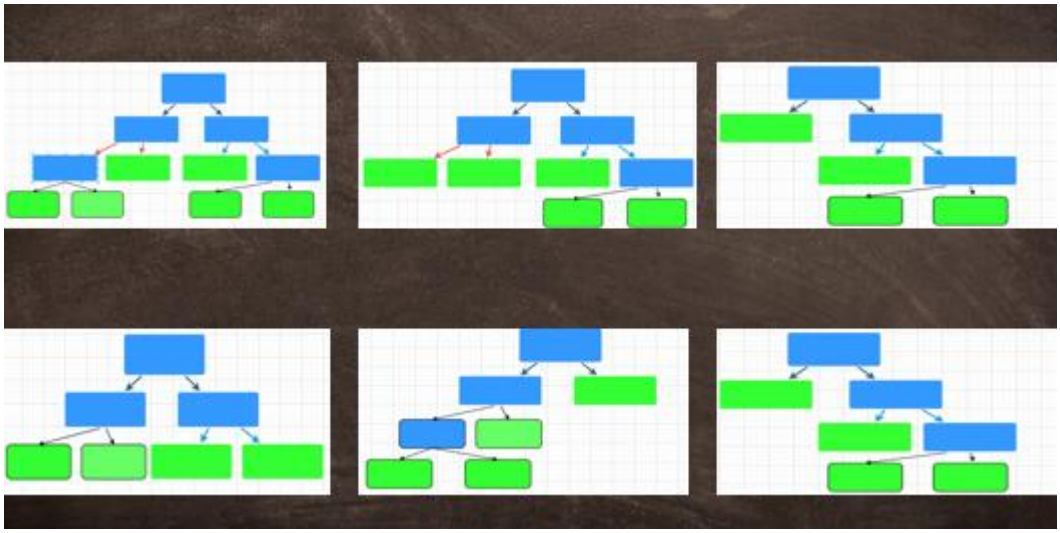
Chest pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
NO	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

- Bootrap Dataset:

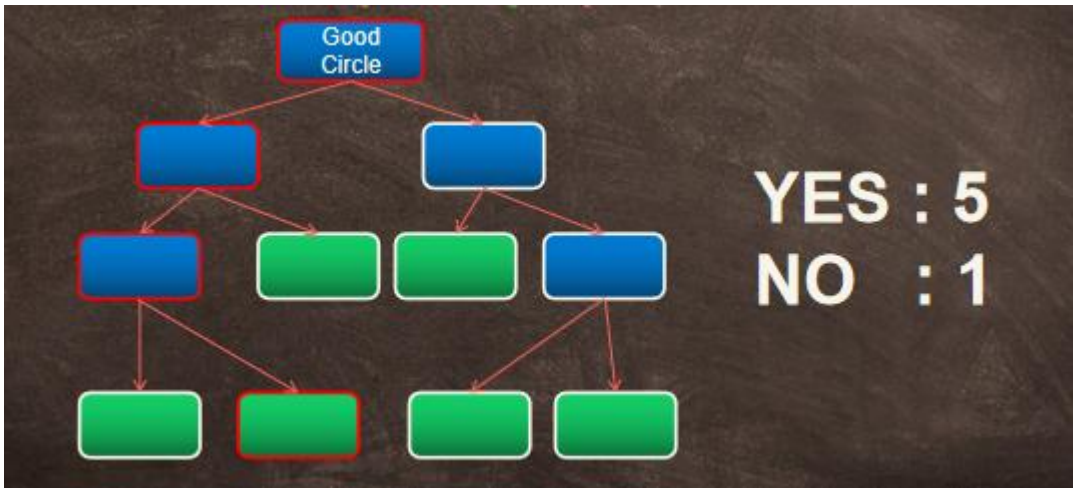
Chest pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes



Chest pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes



Chest pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	



Chest pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	Yes

II. Pipeline

Pipeline là một cách tổ chức và tự động hóa quá trình xử lý dữ liệu và huấn luyện mô hình. Một pipeline bao gồm một chuỗi các bước xử lý dữ liệu và huấn luyện mô hình được thực hiện theo một thứ tự nhất định. Các bước này bao gồm tiền xử lý dữ liệu, chọn đặc trưng, và huấn luyện mô hình.

Dưới đây là một số khái niệm cơ bản liên quan đến pipeline trong machine learning:

- Tiền Xử Lý Dữ Liệu (Data Preprocessing):

Bao gồm các bước như xử lý giá trị bị thiếu, chuyển đổi dữ liệu về định dạng phù hợp, chuẩn hóa dữ liệu, và một số bước xử lý khác để làm sạch dữ liệu.

- Chọn Đặc Trưng (Feature Selection):

Quyết định xem bao nhiêu và những đặc trưng nào sẽ được sử dụng để huấn luyện mô hình. Mục tiêu là giảm chiều của dữ liệu và tăng độ hiệu quả của mô hình.

- Huấn Luyện Mô Hình (Model Training):

Sử dụng dữ liệu đã được tiền xử lý và đã được chọn đặc trưng để huấn luyện mô hình machine learning.

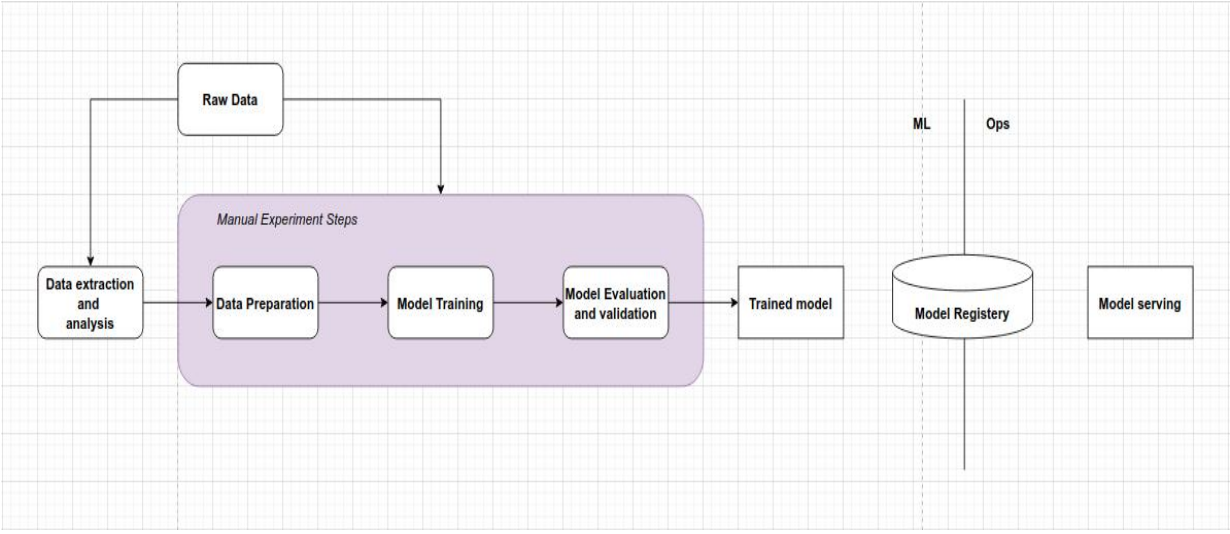
- Đánh Giá Mô Hình (Model Evaluation):

Đánh giá hiệu suất của mô hình trên dữ liệu kiểm thử để đảm bảo mô hình là đáng tin cậy và có khả năng dự đoán tốt trên dữ liệu mới.

- Lưu Trữ và Triển Khai (Saving and Deployment):

Lưu trữ mô hình đã huấn luyện và triển khai nó để sử dụng trong các ứng dụng thực tế.

Sử dụng pipeline giúp đơn giản hóa và tự động hóa quá trình này. Nó giúp duy trì sự chính xác và tái sử dụng mã nguồn một cách dễ dàng, đồng thời giúp quản lý mô hình và quy trình làm việc của mô hình một cách hiệu quả. Công cụ như scikit-learn trong Python cung cấp các công cụ để tạo và quản lý pipeline.



Trong một pipeline machine learning, có ba khối quan trọng là Data Validation, Data Preprocessing, và Feature Engineering, mỗi khối thực hiện một vai trò quan trọng trong việc chuẩn bị dữ liệu và mô hình cho quá trình huấn luyện và đánh giá. Dưới đây là giải thích chi tiết về mỗi khối:

- *Data Validation (Xác nhận Dữ liệu):*

Mục Tiêu: Đảm bảo dữ liệu đầu vào là đầy đủ, chính xác và phù hợp với yêu cầu của mô hình.

Bước Thực Hiện:

Kiểm tra giá trị bị thiếu: Xác định và xử lý giá trị bị thiếu trong dữ liệu.

Kiểm tra định dạng: Xác minh rằng các đặc trưng có định dạng chính xác và phù hợp.

Kiểm tra ngoại lệ: Phát hiện và xử lý ngoại lệ trong dữ liệu.

Kiểm tra tập hợp dữ liệu: Đảm bảo phân phối của tập huấn luyện và kiểm thử là đồng nhất.

- *Data Preprocessing (Tiền Xử Lý Dữ Liệu):*

Mục Tiêu: Chuẩn bị dữ liệu để có thể sử dụng trong mô hình học máy.

Bước Thực Hiện:

Xử lý giá trị bị thiếu: Điền giá trị bị thiếu hoặc loại bỏ các mẫu/cột có giá trị bị thiếu nhiều.

Chuẩn hóa dữ liệu: Chuẩn hóa đặc trưng để chúng có cùng đơn vị và phạm vi giá trị.

Mã hóa Categorical Variables: Chuyển đổi biến phân loại thành dạng số để mô hình có thể hiểu được.

Scaling: Thực hiện scaling các đặc trưng để giảm thiểu ảnh hưởng của biến có phạm vi giá trị lớn.

- *Feature Engineering (Kỹ Thuật Đặc Trưng):*

Mục Tiêu: Tạo ra các đặc trưng mới hoặc biến đổi đặc trưng hiện tại để cung cấp thông tin hữu ích hơn cho mô hình.

Bước Thực Hiện:

Tạo đặc trưng mới: Tạo ra các đặc trưng mới dựa trên sự sáng tạo và hiểu biết về dữ liệu.

Biến đổi đặc trưng: Áp dụng các biến đổi như log, căn bậc hai, để làm mềm dữ liệu.

Loại bỏ đặc trưng không cần thiết: Loại bỏ các đặc trưng không quan trọng hoặc tạo ra độ tương quan cao.

Khi được tổ chức thành một pipeline, các bước trên được xử lý một cách tự động và có thể dễ dàng tái sử dụng cho các dự án khác. Điều này giúp giảm thiểu khả năng mắc lỗi và tăng tính nhất quán của quá trình chuẩn bị dữ liệu và xây dựng mô hình.

III. Triển khai mô hình dự đoán tỷ lệ đột quỵ bằng Random Forest với Pipeline

1. Ngôn ngữ Python

Ngôn ngữ lập trình Python là một ngôn ngữ linh hoạt, dễ đọc và dễ học, được sử dụng rộng rãi trong nhiều lĩnh vực, trong đó bao gồm ứng dụng đặc biệt mạnh mẽ trong lĩnh vực học máy (machine learning). Dưới đây là một tóm tắt về Python và ứng dụng của nó trong học máy:

Tính Chất Chính của Python:

- Dễ Học và Đọc:

Python có cú pháp rất gần với ngôn ngữ tự nhiên, điều này giúp làm giảm độ dốc cho người mới học lập trình.

- Ngôn Ngữ Mở Rộng:

Python là một ngôn ngữ mở rộng, có nghĩa là nó hỗ trợ nhiều phong cách lập trình, bao gồm lập trình hàm, lập trình hướng đối tượng và lập trình thủ tục.

- Cộng Đồng Mạnh Mẽ:

Python có một cộng đồng lớn và đa dạng, cung cấp nhiều thư viện và framework cho mọi mục đích, bao gồm cả học máy.

- Đa Nền Tảng:

Python có thể chạy trên nhiều hệ điều hành khác nhau, giúp dễ dàng triển khai ứng dụng trên nhiều môi trường.

Ứng Dụng của Python trong Học Máy:

- Thư Viện và Framework Học Máy:

Python có những thư viện và framework mạnh mẽ như scikit-learn, TensorFlow, PyTorch, Keras, và một số khác, giúp phát triển và triển khai mô hình học máy một cách dễ dàng.

- Đồ Họa và Trực Quan Hóa:

Thư viện như Matplotlib và Seaborn giúp hiển thị dữ liệu một cách trực quan, giúp hiểu rõ hơn về dữ liệu và kết quả của mô hình.

- Xử Lý Dữ Liệu và Tiền Xử Lý:

Pandas là một thư viện phổ biến giúp xử lý và phân tích dữ liệu. Python cũng cung cấp nhiều công cụ khác như NumPy cho xử lý ma trận và mảng nhanh chóng.

- Ứng Dụng Web và API:

Flask và Django là những framework web phổ biến, giúp triển khai mô hình học máy dưới dạng dịch vụ web hoặc API.

- Học Máy Trong Thực Tế:

Python được sử dụng rộng rãi trong các dự án thực tế về học máy, bao gồm phân loại hình ảnh, dịch ngôn ngữ tự nhiên, dự đoán chuỗi thời gian, và nhiều ứng dụng khác.

Python đã trở thành một trong những ngôn ngữ chính trong cộng đồng học máy và trí tuệ nhân tạo, nhờ vào tính linh hoạt, dễ sử dụng và một hệ sinh thái đa dạng các công cụ hỗ trợ.

2. Công cụ JupyterNotebook

Jupyter Notebook là một môi trường tính toán tương tác và làm việc với mã nguồn mở được sử dụng rộng rãi trong cộng đồng nghiên cứu, giáo dục và phân tích dữ liệu. Dưới đây là một tóm tắt về khái niệm của Jupyter Notebook:

Tính Năng Chính:

- Tương Tác và Linh Hoạt:

Jupyter Notebook cung cấp một môi trường tính toán tương tác, cho phép người dùng thực hiện và kiểm thử mã nguồn từng phần một.

- Hỗ Trợ Nhiều Ngôn Ngữ:

Hỗ trợ nhiều ngôn ngữ lập trình, như Python, R, Julia, và nhiều ngôn ngữ khác.

- Trực Quan Hóa Dữ Liệu:

Cho phép trực quan hóa dữ liệu ngay tại notebook bằng cách sử dụng thư viện như Matplotlib, Seaborn, Plotly.

- Kết Hợp Văn Bản và Mã Nguồn:

Có thể tích hợp cả văn bản được định dạng bằng Markdown và mã nguồn, giúp tạo ra tài liệu tương tác và dễ hiểu.

- Chia Sẻ và Xuất Bản:

Cho phép chia sẻ notebook dễ dàng, có thể xuất ra nhiều định dạng như HTML, PDF, hoặc chuyển đổi thành các định dạng khác.

Cấu Trúc Cơ Bản:

- Cell:

Là đơn vị cơ bản của Jupyter Notebook. Mỗi cell có thể chứa mã nguồn, văn bản hoặc kết quả của mã nguồn.

- Kernel:

Là môi trường thực thi mã nguồn của notebook. Mỗi notebook chạy trên một kernel cụ thể.

- Markdown Cells:

Có thể chứa văn bản được định dạng bằng Markdown để tạo ra tiêu đề, danh sách, đường link, hình ảnh, và nhiều định dạng văn bản khác.

- Code Cells:

Chứa mã nguồn có thể thực thi bởi kernel. Kết quả của mã nguồn sẽ được hiển thị ngay dưới cell.

Sử Dụng Phổ Biến:

- Học Máy và Nghiên Cứu:

Phổ biến trong cộng đồng học máy và nghiên cứu khoa học.

- Giảng Dạy:

Được sử dụng trong giáo dục để giảng dạy lập trình, kết hợp lý thuyết với thực hành.

- Phân Tích Dữ Liệu:

Thường được sử dụng để phân tích và trực quan hóa dữ liệu.

- Báo Cáo Khoa Học:

Dùng để viết và chia sẻ báo cáo nghiên cứu và dự án khoa học.

Jupyter Notebook đã trở thành một công cụ quan trọng trong cộng đồng khoa học dữ liệu và học máy, mang lại tính tương tác và linh hoạt trong quá trình phát triển và trình bày mã nguồn và kết quả.

3. Stroke Prediction Dataset

Kaggle là một nền tảng trực tuyến nổi tiếng cho cộng đồng khoa học dữ liệu và machine learning. Nó cung cấp một môi trường trực tuyến cho các nhà khoa học dữ liệu và nhà phân tích để chia sẻ, thảo luận và thực hiện các dự án khoa học dữ liệu.

Bộ dữ liệu "Stroke Prediction Dataset" mà bạn đưa ra là một ví dụ về tập dữ liệu được chia sẻ trên Kaggle. Dưới đây là một sơ qua về nó:

Dataset: Stroke Prediction Dataset

- Nguồn Gốc:

Bộ dữ liệu này được tạo bởi fedesoriano và được chia sẻ trên Kaggle. Nó được thiết kế để dự đoán nguy cơ đột quỵ dựa trên một số thuộc tính của bệnh nhân.

- Thuộc Tính:

Bộ dữ liệu bao gồm các thuộc tính như tuổi, giới tính, huyết áp, bệnh tim, hôn nhân, loại công việc, loại hút thuốc, loại cư trú, và một số yếu tố khác.

- Mục Tiêu:

Mục tiêu của bộ dữ liệu là dự đoán liệu một người có rủi ro đột quỵ hay không.

- Số Lượng Mẫu:

Bộ dữ liệu có một số lượng mẫu đủ lớn để huấn luyện và kiểm thử mô hình.

- Mục Đích Sử Dụng:

Thường được sử dụng để thực hành và thử nghiệm các mô hình học máy dựa trên dữ liệu y tế.

4. Triển khai mô hình

- Các thư viện cần thiết: Các thư viện được import bao gồm pandas cho xử lý dữ liệu, các thư viện từ scikit-learn cho mô hình hóa và đánh giá mô hình, và các thư viện như matplotlib và seaborn để vẽ biểu đồ.

```
# import thư viện
import pandas as pd # load data
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, mean_squared_error, accuracy_score,
classification_report, precision_score, roc_curve, roc_auc_score, precision_recall_curve
import matplotlib.pyplot as plt
import seaborn as sns
```

- Dữ liệu và tiền xử lý dữ liệu: Dữ liệu từ file CSV được đọc vào DataFrame. Sau đó, kiểm tra và xử lý dữ liệu bị thiếu và loại bỏ cột không cần thiết (cột 'id'). Sau đó, một số cột chứa dữ liệu không phải dạng số được mã hóa thành số nguyên bằng LabelEncoder

```
# loading the data
stroke = pd.read_csv("healthcare-dataset-stroke-data.csv")
```

+ 5 dòng đầu của dataset:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

+ Kiểm tra và xử lý missing data và dropping items:

```
# kiểm tra missing data và dropping item
stroke.isna().sum()
# dropna để loại bỏ cột bị thiếu
stroke = stroke.dropna()
# bỏ cột id
stroke = stroke.drop('id', axis=1)
stroke.isna().sum()
```

+ Các trường dữ liệu sau khi xử lý:

```
gender               0
age                  0
hypertension         0
heart_disease        0
ever_married         0
work_type            0
Residence_type       0
avg_glucose_level    0
bmi                  0
smoking_status       0
stroke               0
dtype: int64
```

+ Theo nguồn dữ liệu, đa số các biến đều mang tính phân loại, có thể sử dụng Label Encoder để gán các nhãn số cho các biến phân loại có tính chất thứ tự.

```
gender              object
age                float64
hypertension         int64
heart_disease        int64
ever_married        object
work_type           object
Residence_type      object
avg_glucose_level  float64
bmi                float64
smoking_status      object
stroke               int64
dtype: object
```

+ Mã hóa nhãn dữ liệu:

```
### Mã hóa nhãn dữ liệu:
le = LabelEncoder()
# các cột cần mã hóa
cols =
['gender','hypertension','heart_disease','ever_married','work_type','Residence_type','smo
king_status','stroke']
# quá trình mã hóa nhãn
stroke[cols] = stroke[cols].apply(lambda col: le.fit_transform(col))
```

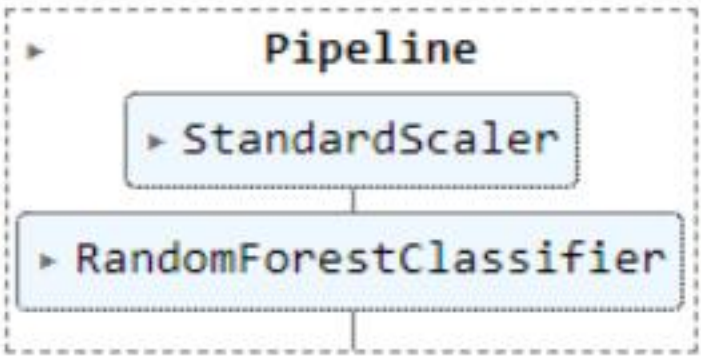
+ 5 dòng dữ liệu đầu sau khi mã hóa:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	1	67.0	0	1	1	2	1	228.69	36.6	1	1
2	1	80.0	0	1	1	2	0	105.92	32.5	2	1
3	0	49.0	0	0	1	2	1	171.23	34.4	3	1
4	0	79.0	1	0	1	3	0	174.12	24.0	2	1
5	1	81.0	0	0	1	2	1	186.21	29.0	1	1

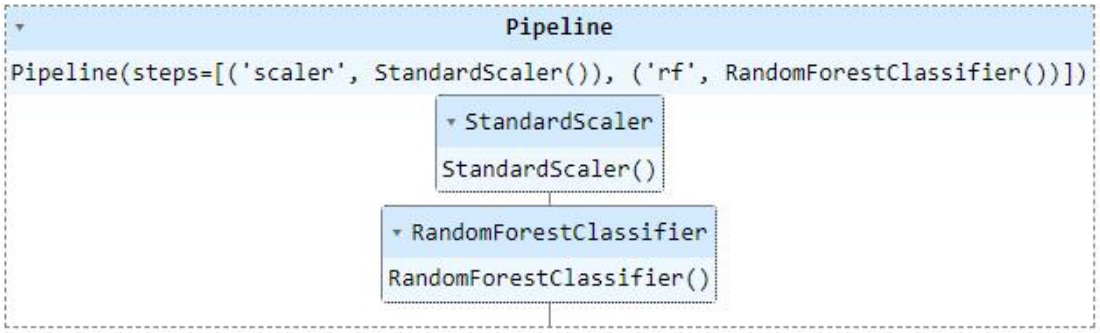
- Sử dụng Pipeline để chia tỷ lệ dữ liệu và áp dụng mô hình RandomForestClassifier.:

```
# Để chia tỷ lệ dữ liệu trước khi đưa vào mô hình RF, chúng ta sử dụng 1 pipeline
# Defining pipeline
# scaler và rf classifier
# pipeline sẽ chia tỷ lệ dữ liệu trước khi đưa vào classifier
pipe = Pipeline([('scaler', StandardScaler()),('rf', RandomForestClassifier())])
```

+ Mô hình Pipeline:



+ Chi tiết mô hình pipeline:



+ Phân chia dữ liệu: Dữ liệu được chia thành tập huấn luyện và tập kiểm thử sử dụng train_test_split.

Trong đó X là tập dữ liệu đặc trưng(ma trận đặc trưng (feature matrix)) và y là tập dữ liệu mục tiêu (vectơ nhãn(label vector))

Tham số test_size xác định tỷ lệ phần trăm của tập kiểm tra trong tập dữ liệu ban đầu.

Tham số random_state xác định hạt giống cho quá trình ngẫu nhiên.

X_train: tập dữ liệu đặc trưng của tập huấn luyện.

X_test: tập dữ liệu đặc trưng của tập kiểm tra.

y_train: tập dữ liệu mục tiêu của tập huấn luyện.

y_test: tập dữ liệu mục tiêu của tập kiểm tra.

```
# Phân chia dữ liệu:
# Xác định X và y
X = stroke.loc[:, 'gender':'smoking_status']
y = stroke.stroke
# chia dữ liệu
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
random_state=42)
```

- Huấn luyện mô hình RandomForest với Pipeline:

```
pipe.fit(X_train, y_train)
```

- Đánh giá mô hình: Mô hình được huấn luyện trên tập huấn luyện và sau đó được đánh giá trên tập kiểm thử.

```
# Đánh giá mô hình
acc = pipe.score(X_test, y_test)
print('Accuracy:', acc)
cm = confusion_matrix(y_test, pipe.predict(X_test))
print('Confusion matrix:\n', cm)
cr= classification_report(y_test, pipe.predict(X_test))
print('Classification report:\n', cr)
```

+ Kết quả đánh giá:

*Accuracy trong machine learning là chỉ số đo lường tần suất thuật toán phân loại một điểm dữ liệu một cách chính xác.

Accuracy: 0.9517990495587237

*Confusion matrix là một phương pháp đánh giá kết quả của những bài toán phân loại với việc xem xét cả những chỉ số về độ chính xác và độ bao quát của các dự đoán cho từng lớp. Confusion matrix thường được biểu diễn dưới dạng một ma trận 2x2 với 4 giá trị True Positive (TP), False Positive (FP), False Negative (FN) và True Negative (TN).

TP: Số lượng các trường hợp mô hình dự đoán đúng và thực tế cũng là đúng.

FP: Số lượng các trường hợp mô hình dự đoán sai và thực tế là đúng.

FN: Số lượng các trường hợp mô hình dự đoán sai và thực tế cũng là sai.

TN: Số lượng các trường hợp mô hình dự đoán đúng và thực tế cũng là sai.

Confusion matrix:

[[1401 0]

[71 1]]

*Classification report: Trong classification report của scikit-learn, *precision* là tỉ lệ giữa số lượng dự đoán đúng positive và tổng số lượng dự đoán positive. *Recall* là tỉ lệ giữa số lượng dự đoán đúng positive và tổng số lượng thực tế positive. *F1-score* là trung bình điều hòa giữa precision và recall. *Support* là số lượng mẫu thực tế trong mỗi class.

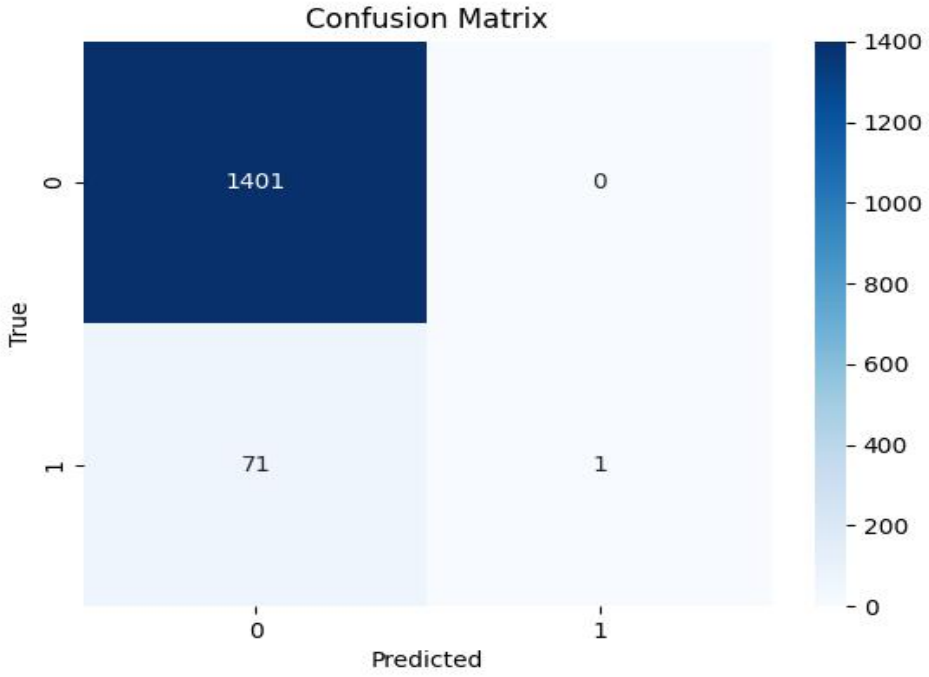
	precision	recall	f1-score	support
0	0.95	1.00	0.98	1401
1	1.00	0.01	0.03	72
accuracy			0.95	1473
macro avg	0.98	0.51	0.50	1473
weighted avg	0.95	0.95	0.93	1473

Macro avg và *weighted avg* là hai chỉ số đánh giá mô hình phân loại trong machine learning. Macro avg là trung bình cộng của precision và recall của từng lớp. Trong khi đó, weighted avg là trung bình cộng của precision và recall của từng lớp với trọng số là tỷ lệ số lượng mẫu của từng lớp so với tổng số lượng mẫu của tất cả các lớp.

- Biểu đồ trực quan:

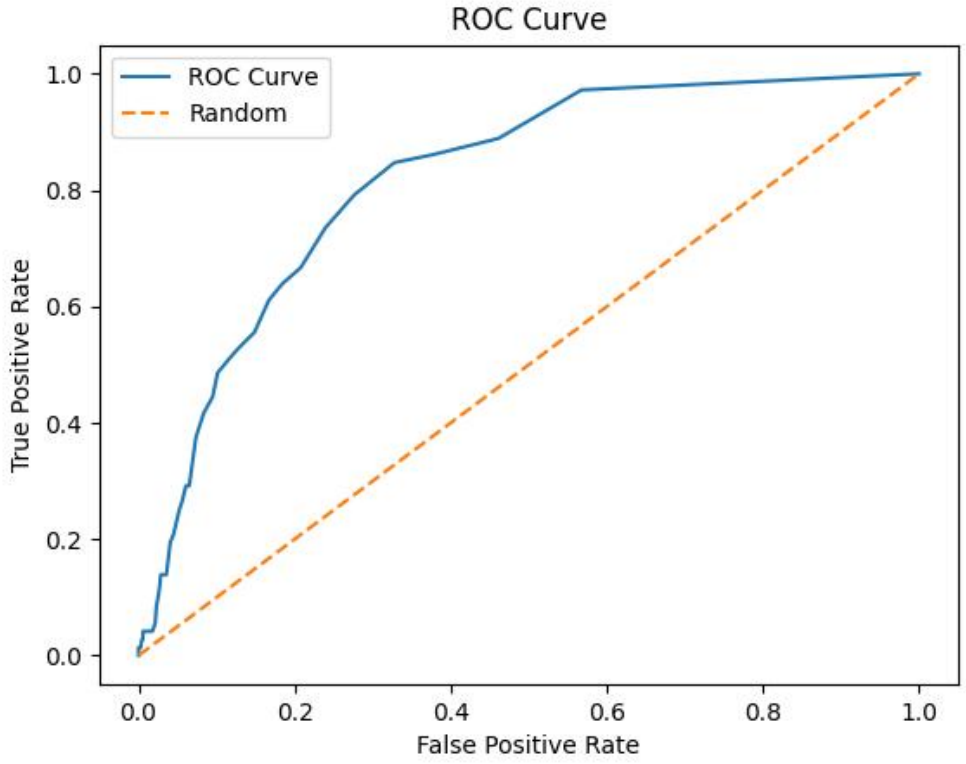
+ Sử dụng thư viện Seaborn để vẽ biểu đồ Confusion Matrix, giúp hiển thị số lượng True Positives, True Negatives, False Positives và False Negatives. Biểu đồ này giúp đánh giá khả năng của mô hình trong việc dự đoán các lớp.

```
# Biểu đồ Confusion Matrix:
sns.heatmap(cm, annot=True, fmt='g', cmap='Blues', xticklabels=['0', '1'],
yticklabels=['0', '1'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```



+ Biểu đồ ROC Curve và tính toán diện tích dưới đường cong (AUC). ROC Curve thường được sử dụng để đánh giá hiệu suất của mô hình phân loại nhị phân dựa trên độ nhạy và độ đặc hiệu. Tính diện tích dưới đường cong (AUC) để đo lường tỷ lệ giữa True Positive Rate và False Positive Rate. Diện tích dưới đường cong (AUC) càng cao, mô hình càng tốt.

```
# Biểu đồ ROC Curve và AUC:
y_probs = pipe.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
plt.plot(fpr, tpr, label='ROC Curve')
plt.plot([0, 1], [0, 1], linestyle='--', label='Random')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
auc = roc_auc_score(y_test, y_probs)
print('AUC:', auc)
```

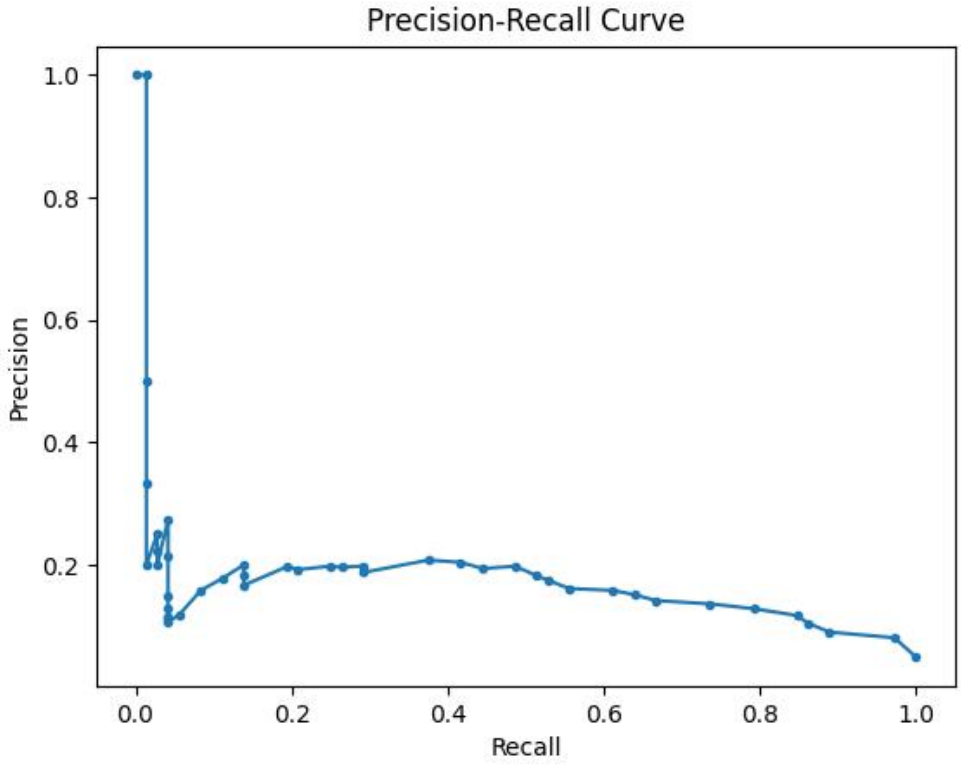


AUC: 0.8164307240859703

+ Biểu đồ Precision-Recall Curve dùng để đánh giá cân bằng giữa precision và recall của mô hình. Điều này quan trọng khi mô hình phải đối mặt với một môi trường mà số lượng False Positives càng ít càng tốt.

```
# Biểu đồ Precision-Recall Curve:
precision_curve, recall_curve, _ = precision_recall_curve(y_test, y_probs)

plt.plot(recall_curve, precision_curve, marker='.')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.show()
```


Precision-Recall Curve
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0