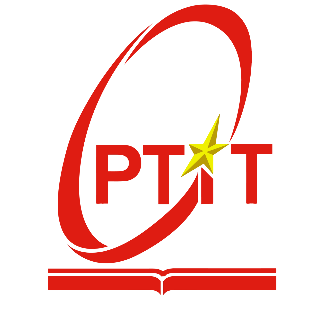
 **HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH**

---------------🕮--------------



**ĐỒ ÁN GIỮA KỲ**

**MÔN KHO DỮ LIỆU VÀ KHAI PHÁ**

**DỮ LIỆU**

**GIẢNG VIÊN HƯỚNG DẪN: NGUYỄN NGỌC DUY**

**NHÓM 1**

**Đề tài : Đặc tả dữ liệu mẫu “diabetes.arff” của Weka. Chọn phương pháp phù hợp để khai phá dữ này.**

**N21DCCN101 – Vũ Quốc Hoàng Anh**

**N21DCCN016 - Triệu Quốc Đạt**

**Thành phố Hồ Chí Minh**

MỤC LỤC

[LỜI NÓI ĐẦU 3](#_Toc6068)

[CHƯƠNG I: GIỚI THIỆU CHUNG 1](#_Toc27273)

[3.1 Nguồn gốc dữ liệu 2](#_Toc13727)

[3.2 Mục tiêu của thu thập dữ liệu 2](#_Toc18258)

[CHƯƠNG II: ĐẶT TẢ DỮ LIỆU 3](#_Toc16457)

[1. Phân tích dữ liệu 3](#_Toc30305)

[2. Mô tả dữ liệu 8](#_Toc12179)

[3. Trực quan hóa dữ liệu 9](#_Toc30273)

[4. Tiền xử lý dữ liệu: 11](#_Toc32091)

[Nhận xét chung: 13](#_Toc205)

[CHƯƠNG III: CÔNG CỤ VÀ PHƯƠNG PHÁP KHAI PHÁ DỮ LIỆU 14](#_Toc28067)

[1. Giới thiệu công cụ 14](#_Toc15695)

[2. Phương pháp 18](#_Toc22705)

[Tiền xử lý dữ liệu 27](#_Toc16524)

[CHƯƠNG IV: ĐÁNH GIÁ MÔ HÌNH 27](#_Toc3941)

[1. Mô hình kNN trước khi chuẩn hóa. 27](#_Toc12880)

[2. Mô hình kNN sau khi chuẩn hóa. 29](#_Toc29469)

[3. Nhận xét mô hình kNN. 32](#_Toc27701)

[4. Kết luận. 33](#_Toc26081)

[CHƯƠNG V: XÂY DỰNG ỨNG DỤNG 34](#_Toc17332)

[1. Giới thiệu về Flask và React 34](#_Toc7332)

[2. Code chương trình 34](#_Toc459)

[Chức năng chính của code 34](#_Toc27872)

[CHƯƠNG VI: KẾT LUẬN 43](#_Toc23734)

**LỜI NÓI ĐẦU**

Bài tiểu luận không chỉ dừng lại ở việc phân tích và đặc tả dữ liệu mà còn lựa chọn phương pháp khai phá phù hợp, cụ thể là thuật toán k-Nearest Neighbors (kNN). Đồng thời, chúng tôi cũng tiến hành đánh giá hiệu suất của mô hình trước và sau khi chuẩn hóa dữ liệu nhằm tìm ra giải pháp tối ưu nhất. Trên cơ sở đó, một công cụ giao diện được xây dựng giúp người dùng có thể nhập các triệu chứng và nhận được kết luận về nguy cơ mắc bệnh tiểu đường, hỗ trợ việc theo dõi sức khỏe cá nhân cũng như hỗ trợ bác sĩ trong quá trình sàng lọc bệnh nhân.

Trong quá trình thực hiện bài tiểu luận, chúng em đã nhận được sự hướng dẫn tận tình từ Thầy Nguyễn Ngọc Duy. Thầy đã cung cấp những kiến thức bổ ích, giúp chúng em hiểu sâu hơn về các phương pháp khai phá dữ liệu và ứng dụng trong thực tế. Chúng em xin gửi lời cảm ơn chân thành đến Thầy vì những chỉ dẫn quý báu đó. Đồng thời, chúng em cũng xin gửi lời cảm ơn đến các thầy cô bộ môn đã tạo điều kiện để chúng em có cơ hội nghiên cứu và hoàn thành bài tiểu luận này.

Dù đã có nhiều nỗ lực trong quá trình thực hiện, nhưng chắc chắn bài tiểu luận không thể tránh khỏi những thiếu sót. Chúng em rất mong nhận được ý kiến đóng góp từ Thầy Cô và các bạn để bài nghiên cứu được hoàn thiện hơn.

Nhóm xin chân thành cảm ơn!

# CHƯƠNG I: GIỚI THIỆU CHUNG

1. **Giới thiệu kho dữ liệu và khai phá dữ liệu:**

Kho dữ liệu (Data Warehouse) và khai phá dữ liệu (Data Mining) là hai lĩnh vực quan trọng trong khoa học dữ liệu và trí tuệ nhân tạo. Kho dữ liệu là mô hình tổng hợp, lưu trữ dữ liệu từ nhiều nguồn khác nhau, hỗ trợ việc phân tích và ra quyết định. Trong khi đó, khai phá dữ liệu tập trung vào việc tìm kiếm các mẫu hình, xu hướng và kiến thức ẩn giấu từ kho dữ liệu bằng các kỹ thuật học máy.

Khai phá dữ liệu đóng vai trò quan trọng trong nhiều lĩnh vực như tài chính, y tế, marketing và khoa học. Trong y tế, khai phá dữ liệu giúp phân tích hàng loạt dữ liệu bệnh nhân để phát hiện nguyên nhân gây bệnh, đề xuất phương pháp chẩn đoán và hỗ trợ quyết định y khoa.

1. **Giới thiệu đề tài thực hiện:**

Trong bối cảnh y học hiện đại, bệnh tiểu đường ngày càng trở thành một vấn đề nghiêm trọng ảnh hưởng đến sức khỏe con người trên toàn cầu. Việc chẩn đoán sớm và chính xác bệnh tiểu đường có ý nghĩa quan trọng trong việc kiểm soát và điều trị, giúp giảm thiểu các biến chứng nguy hiểm. Tuy nhiên, các phương pháp chẩn đoán truyền thống thường phụ thuộc nhiều vào kinh nghiệm của bác sĩ và có thể không khai thác được hết thông tin từ dữ liệu bệnh nhân.

Kho dữ liệu và khai phá dữ liệu đóng vai trò quan trọng trong việc hỗ trợ y học, đặc biệt trong việc xây dựng các mô hình dự đoán bệnh tật. Bằng cách phân tích dữ liệu y tế, các thuật toán học máy có thể phát hiện ra những mẫu tiềm ẩn giúp cải thiện khả năng dự đoán bệnh chính xác hơn. Trong nghiên cứu này, chúng em tập trung vào việc khai thác bộ dữ liệu "diabetes.arff" của Weka nhằm xây dựng một công cụ hỗ trợ chẩn đoán nguy cơ mắc bệnh tiểu đường.

Việc lựa chọn thuật toán k-Nearest Neighbors (kNN) xuất phát từ khả năng đơn giản nhưng hiệu quả của nó trong việc phân loại dữ liệu y khoa. Để nâng cao độ chính xác của mô hình, chúng em cũng nghiên cứu tác động của việc chuẩn hóa dữ liệu đối với hiệu suất của kNN. Cuối cùng, từ mô hình đạt hiệu suất tốt nhất, chúng em xây dựng một ứng dụng giao diện giúp người dùng có khả năng nhận được kết luận về nguy cơ mắc bệnh tiểu đường. Công cụ này không chỉ hỗ trợ cá nhân trong việc tự đánh giá sức khỏe mà còn có tiềm năng trở thành một hệ thống hỗ trợ bác sĩ trong quá trình chẩn đoán.

1. **Giới thiệu chung về bộ dữ liệu diabetes.arff của Weka:**

3.1 Nguồn gốc dữ liệu

Tập dữ liệu Diabetes là một trong những tập dữ liệu y tế phổ biến nhất, thường được sử dụng để nghiên cứu và phát triển các mô hình dự đoán bệnh tiểu đường. Đây là bộ dữ liệu tiêu chuẩn trong lĩnh vực y học, học máy và trí tuệ nhân tạo, đặc biệt là trong các bài toán phân loại y tế.

Dữ liệu được thu thập bởi Viện Quốc gia về Bệnh Tiểu đường, Tiêu hóa & Bệnh Thận của Hoa Kỳ (National Institute of Diabetes and Digestive and Kidney Diseases - NIDDK). Tập dữ liệu này tập trung vào nhóm phụ nữ người Mỹ gốc Pima, một cộng đồng có tỷ lệ mắc bệnh tiểu đường cao hơn mức trung bình so với các nhóm dân số khác.

Dữ liệu được thu thập trong nhiều năm thông qua các cuộc khảo sát y tế và xét nghiệm lâm sàng. Các thông số trong tập dữ liệu được đo lường bằng các phương pháp y tế tiêu chuẩn, đảm bảo tính chính xác và đáng tin cậy.

3.2 Mục tiêu của thu thập dữ liệu

Mục tiêu chính của tập dữ liệu Diabetes là phát triển mô hình dự đoán khả năng mắc bệnh tiểu đường dựa trên các thông số y tế và thông tin sinh học của bệnh nhân. Cụ thể:

Xác định các yếu tố chính có liên quan đến bệnh tiểu đường.

Hỗ trợ bác sĩ và nhà nghiên cứu trong việc phân tích dữ liệu y tế.

Cung cấp bộ dữ liệu tiêu chuẩn để huấn luyện các thuật toán học máy, phục vụ dự đoán bệnh tật.

Người Mỹ gốc Pima có tỷ lệ mắc bệnh tiểu đường loại 2 cao hơn đáng kể so với các nhóm dân số khác. Do đó, nghiên cứu về nhóm dân số này có thể giúp tìm ra các yếu tố nguy cơ tiềm ẩn, từ đó áp dụng các biện pháp phòng ngừa và điều trị hiệu quả hơn.

1. **Mục tiêu đề tài:**

Mục tiêu của đề tài này là xây dựng một công cụ hỗ trợ chẩn đoán nguy cơ tiểu đường dựa trên bộ dữ liệu "diabetes.arff" thông qua việc thực hiện các bước từ đặc tả dữ liệu, lựa chọn phương pháp khai phá, đánh giá hiệu suất mô hình và triển khai ứng dụng thực tế. Đầu tiên, việc đặc tả dữ liệu sẽ giúp hiểu rõ cấu trúc, ý nghĩa của các thuộc tính và mối liên hệ giữa chúng, từ đó tạo nền tảng cho việc xây dựng mô hình dự đoán. Tiếp theo, thuật toán k-Nearest Neighbors (kNN) được lựa chọn do tính đơn giản nhưng hiệu quả trong việc phân loại dữ liệu y khoa, giúp mô hình có khả năng học tốt từ dữ liệu mẫu.

Bên cạnh đó, đề tài tập trung vào việc đánh giá hiệu suất của mô hình trước và sau khi chuẩn hóa dữ liệu để kiểm chứng tác động của quá trình tiền xử lý đối với kết quả dự đoán. Các chỉ số như độ chính xác (Accuracy), độ nhạy (Recall), độ chính xác của dự đoán dương tính (Precision) và F1-score sẽ được sử dụng để phân tích mức độ cải thiện của mô hình. Cuối cùng, từ mô hình kNN đã được tối ưu, một giao diện ứng dụng sẽ được xây dựng để giúp người nhận được kết quả đánh giá nguy cơ mắc bệnh tiểu đường từ các chịu chứng. Công cụ này không chỉ giúp người dùng dễ dàng tiếp cận thông tin sức khỏe cá nhân mà còn hỗ trợ các chuyên gia y tế trong việc sàng lọc bệnh nhân có nguy cơ cao, góp phần nâng cao chất lượng chẩn đoán và điều trị bệnh tiểu đường.

# CHƯƠNG II: ĐẶT TẢ DỮ LIỆU

1. **Phân tích dữ liệu**

Bộ dữ liệu Diabetes bao gồm 768 mẫu, mỗi mẫu đại diện cho một bệnh nhân nữ người Mỹ gốc Pima. Mỗi dòng dữ liệu chứa 9 thuộc tính, trong đó 8 thuộc tính đầu vào và 1 thuộc tính đầu ra (nhãn phân loại).

Danh sách các thuộc tính

**1.preg**

Mô tả chi tiết:Số lần mang thai của bệnh nhân

Đơn vị:Số lần

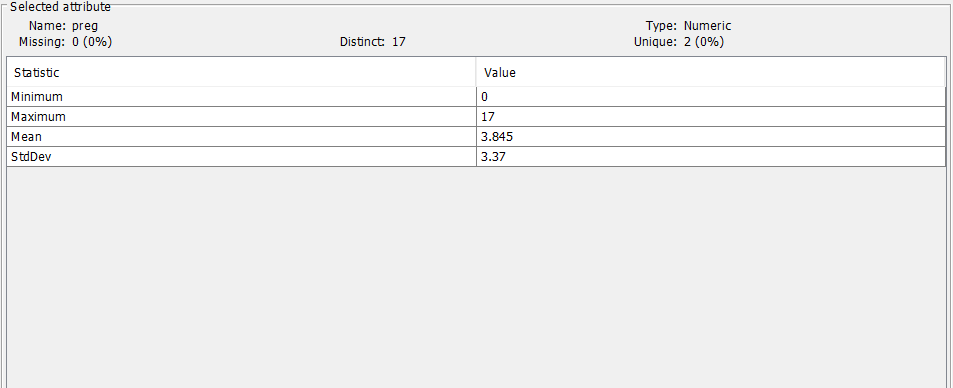
Loại dữ liệu:Số nguyên

Gía trị nhỏ nhất:0

Gía trị lớn nhất:17

Gía trị trung bình:3.845

Độ lệch chuẩn:3.37



*Hình 1:Thông số preg*

**2.plas**

Mô tả chi tiết:Nồng độ glucose trong máu sau 2 giờ làm nghiệm pháp dung nạp glucose

Đơn vị:mg/dL

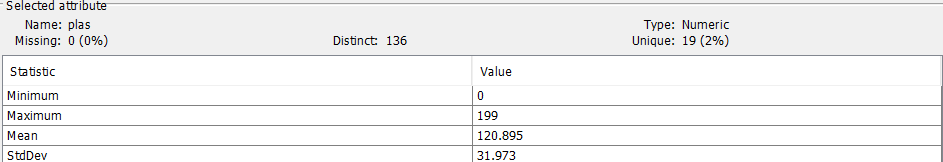
Loại dữ liệu:Số thực

Gía trị nhỏ nhất:0

Gía trị lớn nhất:199

Gía trị trung bình:120.895

Độ lệch chuẩn:31.973



*Hình 2: Thông số plas*

**3.pres**

Mô tả chi tiết:Huyết áp tâm thu

Đơn vị:mmHg

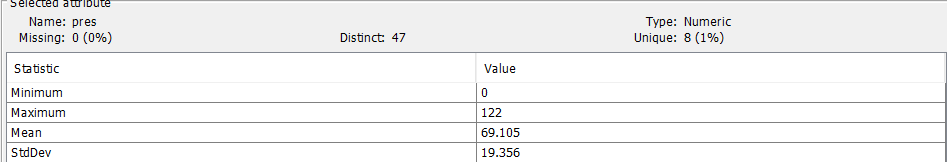
Loại dữ liệu:Số thực

Gía trị nhỏ nhất:0

Gía trị lớn nhất:122

Gía trị trung bình:69.105

Độ lệch chuẩn:19.356



*Hình 3: Thông số pres*

**4.skin**

Mô tả chi tiết:Độ dày lớp da gấp dưới bắp tay (đo bằng thước kẹp)

Đơn vị:mm

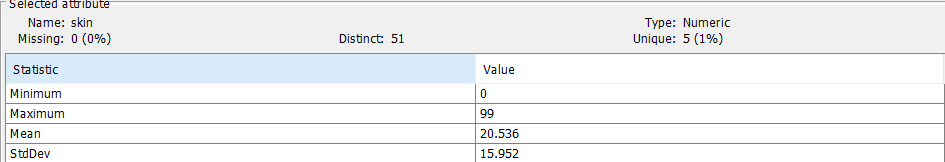
Loại dữ liệu:Số thực

Gía trị nhỏ nhất:0

Gía trị lớn nhất:99

Gía trị trung bình:20.536

Độ lệch chuẩn:15.952



*Hình 4: Thông số skin*

**5.insu**

Mô tả chi tiết:Nồng độ insulin huyết thanh sau 2 giờ

Đơn vị:mu U/ml

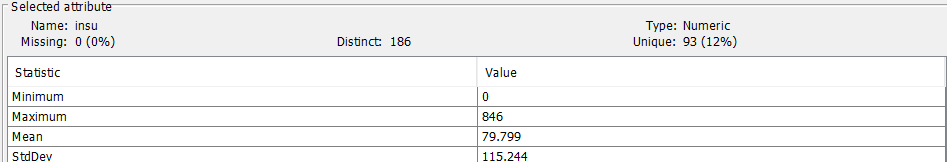
Loại dữ liệu:Số thực

Gía trị nhỏ nhất:0

Gía trị lớn nhất:846

Gía trị trung bình:79.799

Độ lệch chuẩn:115.244



*Hình 5: Thông số insu*

**6.mass**

Mô tả chi tiết:Chỉ số khối cơ thể (BMI - Body Mass Index)

Đơn vị:kg/m²

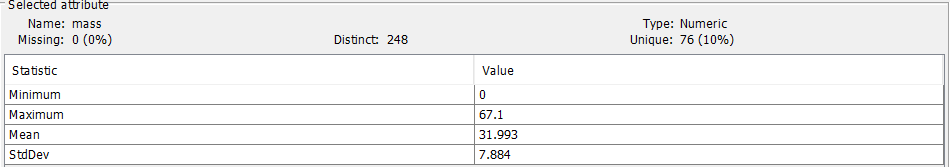
Loại dữ liệu:Số thực

Gía trị nhỏ nhất:0

Gía trị lớn nhất:67.1

Gía trị trung bình:31.993

Độ lệch chuẩn:7.884



*Hình 6: Thông số mass*

**7.pedi**

Mô tả chi tiết:Chỉ số tiền sử bệnh tiểu đường trong gia đình (Diabetes Pedigree Function)

Đơn vị:không

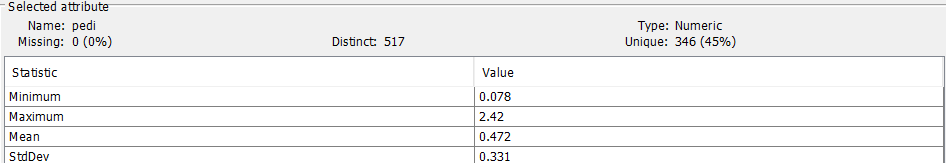
Loại dữ liệu:Số thực

Gía trị nhỏ nhất:0.078

Gía trị lớn nhất:2.42

Gía trị trung bình:0.472

Độ lệch chuẩn:0.331



*Hình 7: Thông số pedi*

**8.age**

Mô tả chi tiết:Tuổi của bệnh nhân

Đơn vị:Năm

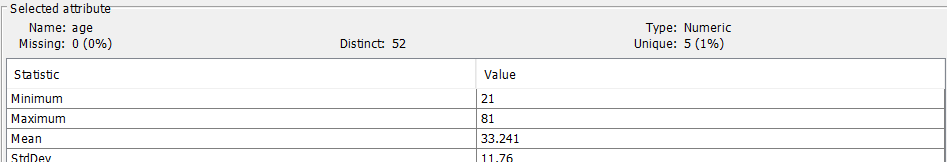
Loại dữ liệu:Số nguyên

Gía trị nhỏ nhất:21

Gía trị lớn nhất:81

Gía trị trung bình:33.241

Độ lệch chuẩn:11.76



*Hình 8: Thông số age*

**9.class**

Mô tả chi tiết:Nhãn phân loại: 0 - Không mắc bệnh, 1 - Mắc bệnh

Đơn vị:không

Loại dữ liệu:số nguyên

Gía trị nhỏ nhất:0

Gía trị lớn nhất:1

1. **Mô tả dữ liệu**

**Thuộc tính đầu vào (biến độc lập)**

**1.preg**

- Số lần bệnh nhân đã mang thai.

- Liên quan đến nguy cơ mắc bệnh tiểu đường thai kỳ.

**2.plas**

- Đo nồng độ glucose trong máu sau khi uống dung dịch glucose.

- Đây là chỉ số quan trọng trong chẩn đoán tiểu đường.

- Giá trị ≥ 200 mg/dL có thể là dấu hiệu của bệnh tiểu đường.

**3.pres**

- Huyết áp đo khi tim co bóp.

- Tăng huyết áp là một yếu tố nguy cơ của bệnh tiểu đường.

**4.Skin**

- Được đo bằng thước kẹp để ước lượng lượng mỡ dưới da.

- Giá trị cao có thể liên quan đến béo phì và nguy cơ tiểu đường.

**5.Insu**

- Đo khả năng tiết insulin của cơ thể.

- Giá trị bằng 0 có thể do dữ liệu bị thiếu.

**6.Mass**

- Tính theo công thức: BMI = cân nặng (kg) / (chiều cao (m)²).

- BMI ≥ 30 được coi là béo phì, làm tăng nguy cơ mắc tiểu đường.

**7.Pedi**

- Đánh giá nguy cơ mắc bệnh dựa trên lịch sử gia đình.

- Giá trị càng cao, nguy cơ mắc bệnh càng lớn.

**8.Age**

- Tuổi càng cao, nguy cơ mắc bệnh tiểu đường càng lớn.

- Người trên 45 tuổi có nguy cơ cao hơn.

**Thuộc tính đầu ra (biến phụ thuộc - nhãn phân loại)**

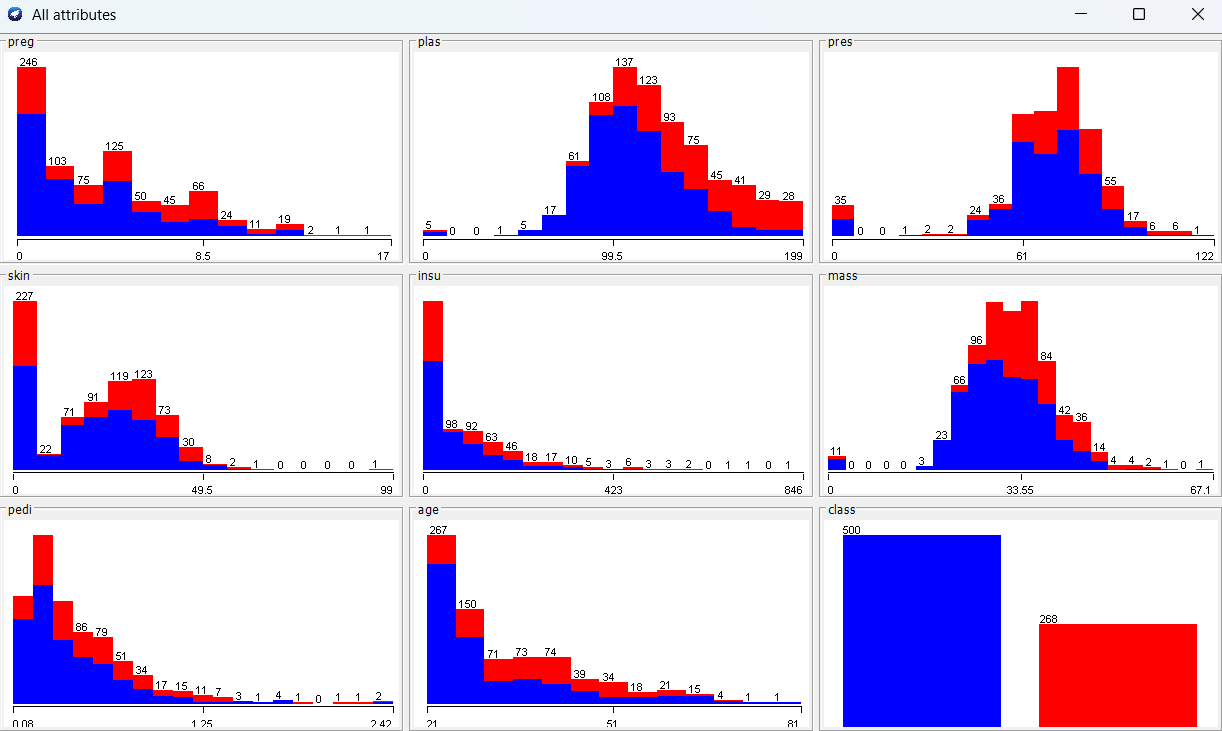
**9.Class(nhãn)**

0: Không mắc bệnh.

1: Mắc bệnh tiểu đường.

Mô hình học máy sẽ dự đoán nhãn này dựa trên các thuộc tính đầu vào.

1. **Trực quan hóa dữ liệu**

****

*Hình 9: Histogram phân bố dữ liệu*

Hình ảnh hiển thị các biểu đồ histogram của các thuộc tính khác nhau trong tập dữ liệu, với hai nhóm dữ liệu được phân biệt bằng hai màu:

Màu xanh (blue): Biểu thị nhóm dữ liệu có nhãn không mắc bệnh.

Màu đỏ (red): Biểu thị nhóm dữ liệu có nhãn mắc bệnh.

Dưới đây là phân tích chi tiết một số thuộc tính chính:

1.**preg** (số lần mang thai): Phân bố lệch phải, phần lớn giá trị nhỏ hơn 8.5, nhưng có một số trường hợp ngoại lệ cao hơn 10.

2.**plas** (nồng độ glucose trong máu): Phân bố gần với chuẩn, tập trung quanh khoảng 100. Nhóm mắc bệnh (màu đỏ) có xu hướng giá trị cao hơn.

3.**pres** (huyết áp): Phân bố gần chuẩn với tâm khoảng 61. Nhóm mắc bệnh có xu hướng giá trị cao hơn so với nhóm còn lại.

4.**skin** (độ dày nếp gấp da): Phân bố lệch phải, với phần lớn giá trị dưới 50.

5.**insu** (nồng độ insulin): Phân bố rất lệch phải, với nhiều giá trị thấp (dưới 100) và một số ngoại lệ rất lớn (hơn 400).

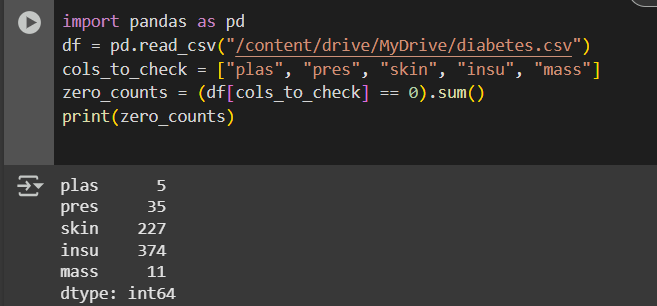
6.**mass** (BMI - chỉ số khối cơ thể): Phân bố gần chuẩn với tâm khoảng 33.5. Nhóm mắc bệnh có xu hướng giá trị cao hơn một chút.

7.**pedi** (chỉ số di truyền tiểu đường): Phân bố lệch phải, phần lớn tập trung dưới 1.

8.**age** (tuổi): Phân bố lệch phải, với phần lớn giá trị dưới 50. Nhóm mắc bệnh có xu hướng tuổi cao hơn.

9.**class** (nhãn phân loại): Có 500 mẫu thuộc nhóm không mắc bệnh (màu xanh) và 268 mẫu thuộc nhóm mắc bệnh (màu đỏ).

1. **Tiền xử lý dữ liệu:**

****

*Hình 10: Tìm số lượng giá trị 0 của các cột*

Vấn đề:

|  |  |
| --- | --- |
| Cột dữ liệu | Số lượng giá trị 0 |
| Glucose (plas) | 5 |
| Blood Pressure (pres) | 35 |
| Skin Thickness (skin) | 227 |
| Insulin (insu) | 374 |
| BMI (mass) | 11 |

**Glucose (plas)**: Không thể là 0 vì mọi người đều có đường huyết.

**Blood Pressure (pres)**: Huyết áp không thể là 0.

**Skin Thickness (skin)**: Không thể có độ dày da bằng 0.

**Insulin (insu)**: Nếu bằng 0, có thể là do chưa đo hoặc dữ liệu bị thiếu.

**BMI (mass)**: Không thể bằng 0 vì mọi người đều có khối lượng cơ thể.

Tất nhiên giá trị = 0 vẫn còn những nguyên nhân khác như không có dữ liệu,máy hư,…

Để thay thế các giá trị này, nhóm em sử dụng KNN Imputer thay vì các phương pháp như trung bình hoặc trung vị, vì:

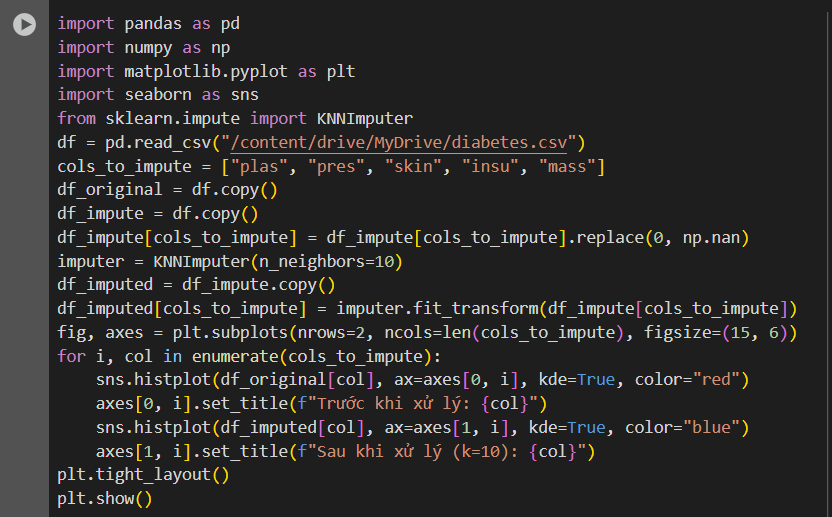
+ Nhiều giá trị 0 trong một số cột quan trọng: Một số cột như insulin, skin thickness có quá nhiều giá trị bằng 0, cho thấy đây có thể là dữ liệu bị thiếu thay vì giá trị thực sự bằng 0. Nếu thay thế bằng trung bình hoặc trung vị, dữ liệu có thể bị bóp méo.

+ Phản ánh đúng mối quan hệ giữa các đặc điểm: KNN Imputer dựa vào các điểm dữ liệu có đặc điểm tương tự để điền giá trị còn thiếu, giúp bảo toàn cấu trúc và mối tương quan giữa các biến.

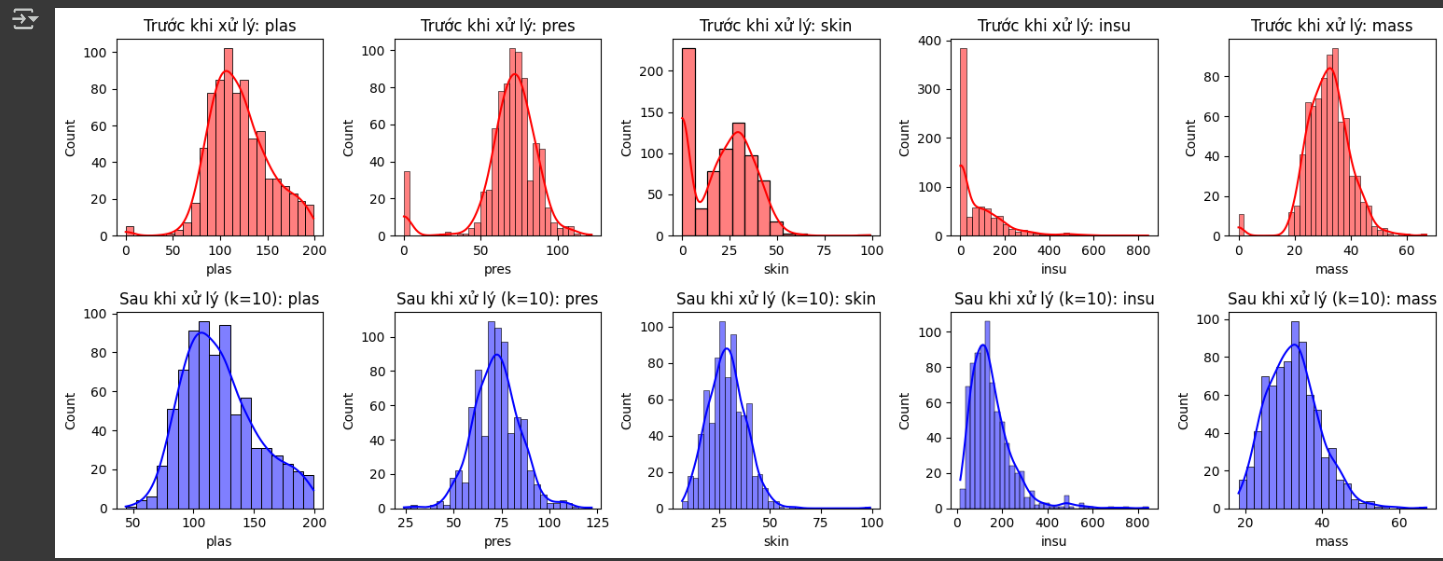
+ Thực tế hơn trong y khoa: Các chỉ số như huyết áp, insulin không thể bằng 0. Việc ước lượng dựa trên bệnh nhân có đặc điểm tương tự giúp dữ liệu sát với thực tế lâm sàng hơn, tránh sai lệch trong phân tích.

+ Bảo toàn phân phối dữ liệu: Trung bình hoặc trung vị có thể làm giảm tính đa dạng và gây lệch phân phối dữ liệu, trong khi KNN Imputer giúp giữ nguyên đặc điểm phân phối của dữ liệu gốc.

+ Đây là chương trình sử dụng thuật toán KNN Imputer (k=10) để xử lý các giá trị thiếu trong dữ liệu diabetes.csv. Các cột chứa giá trị 0 sẽ được thay thế bằng giá trị được nội suy từ k-láng giềng gần nhất.



*Hình 11: Xử lý giá trị thiếu bằng KNN Imputer*



*Hình 12: Dữ liệu trước và sau khi xử lý*

**Nhận xét chung:**

1.Các cột có giá trị 0 nhiều ban đầu như skin, insu có phân bố bị lệch do nhiều giá trị bằng 0.

2.Sau khi xử lý bằng KNN (k=10):

+ Các phân bố trở nên mượt hơn, không còn đỉnh nhọn bất thường do giá trị 0.

+ Đặc biệt, skin và insu đã có dạng phân bố hợp lý hơn, gần với phân bố chuẩn.

+ plas, pres, mass không thay đổi quá nhiều, cho thấy KNN không làm biến dạng dữ liệu gốc.

# CHƯƠNG III: CÔNG CỤ VÀ PHƯƠNG PHÁP KHAI PHÁ DỮ LIỆU

1. **Giới thiệu công cụ**



Hình 13: Công cụ Python

Python là ngôn ngữ chính được sử dụng để xử lý và khai phá dữ liệu diabetes.arff.

Thư viện hỗ trợ

- **scipy.io.arff** hoặc **liac-arff**: Dùng để đọc dữ liệu từ file .arff.

- **pandas**: Hỗ trợ chuyển đổi dữ liệu .arff thành DataFrame để dễ xử lý.

- **scikit-learn (sklearn)**: Thư viện chính để triển khai thuật toán KNN, bao gồm:

+ **KNNImputer**: sử dụng để **điền các giá trị bị thiếu (missing values) trong dữ liệu** bằng thuật toán **K-Nearest Neighbors (KNN)**.

+ **KNeighborsClassifier (phân loại)**

+ **train\_test\_split**: Chia dữ liệu thành tập huấn luyện và kiểm tra.

+ **StandardScaler**: Chuẩn hóa dữ liệu đầu vào để tăng hiệu suất mô hình.

+ **accuracy\_score**: Đánh giá độ chính xác của mô hình.

+ **confusion\_matrix**: Xây dựng ma trận nhầm lẫn để đánh giá kết quả phân loại

+ **classification\_report**:Báo cáo chi tiết về precision, recall, F1-score

- **matplotlib** và **seaborn**: Hỗ trợ trực quan hóa dữ liệu.

- **Flask**: Framework Python để xây dựng API RESTful.

- **Flask-CORS**: Hỗ trợ CORS (Cross-Origin Resource Sharing), cho phép ứng dụng giao tiếp với frontend.

Công cụ hỗ trợ

**Google Colab / Jupyter Notebook .**

**Postman / cURL** (để gửi request đến API Flask và kiểm tra kết quả dự đoán).

**Python Virtual Environment (venv)** hoặc **Conda** (để quản lý thư viện khi chạy trên server thực tế).

**1.1 Tích hợp Machine Learning Algorithms:**

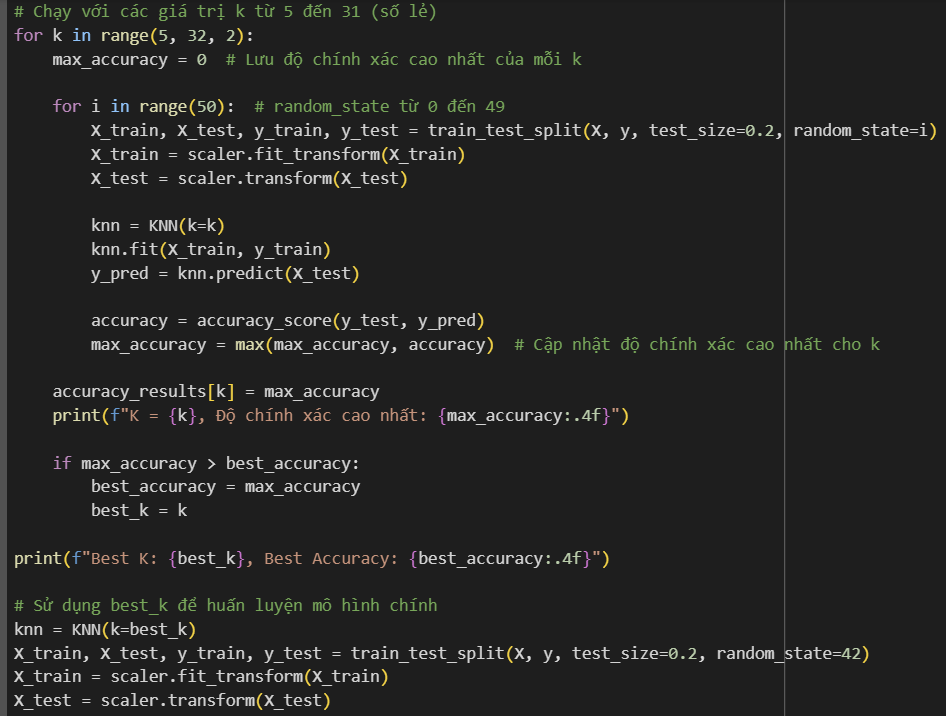
Classification (Phân loại): Mô hình phân loại dữ liệu vào các nhóm hoặc lớp khác nhau, dùng để dự đoán bệnh tiểu đường (có/không),với hai nhãn: 0 (Không bị bệnh) và 1 (Bị bệnh).

Deep Learning (Học Sâu): được sử dụng để huấn luyện mô hình phân loại dữ liệu về bệnh tiểu đường, với cách tiếp cận bằng Mạng Neural Network (NN).

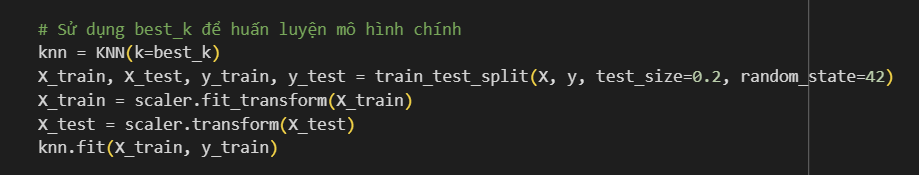
**1.2 Tính linh hoạt:**

Model Selection (Chọn Mô hình): Người dùng lựa chọn thuật toán K-Nearest Neighbors (KNN) để thực hiện bài toán phân loại bệnh tiểu đường, mã nguồn thử nghiệm với nhiều giá trị của K (từ 5 đến 31, số lẻ) để tìm ra mô hình tốt nhất.

Hyperparameter Tuning (Tinh chỉnh Siêu tham số): giá trị K (số lượng láng giềng gần nhất) được thử nghiệm từ 5 đến 31. Với mỗi giá trị K, thuật toán được huấn luyện và kiểm tra để tìm độ chính xác cao nhất, điều này thể hiện quá trình tinh chỉnh siêu tham số để cải thiện độ chính xác.



*Hình 14: Lặp qua nhiều giá trị của k để tìm giá trị tối ưu*

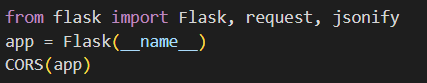


*Hình 15: Huấn luyện mô hình với giá trị k tối ưu*

Performance Evaluation (Kiểm tra Hiệu suất): Sử dụng accuracy\_score để đo độ chính xác của mô hình.Thực hiện train-test split (80% train - 20% test) với nhiều giá trị random\_state từ 0 đến 49 để đảm bảo mô hình ổn định,tính độ chính xác cao nhất cho từng giá trị K để chọn mô hình tối ưu nhất.

Data Preprocessing : Sử dụng StandardScaler để chuẩn hóa đặc trưng (feature scaling), giúp thuật toán KNN hoạt động hiệu quả hơn.

Model Deployment (Triển khai Mô hình với Flask): Flask API được thiết lập (app = **Flask(\_\_name\_\_)**), có thể mở rộng để phục vụ dự đoán trực tuyến.CORS (Cross-Origin Resource Sharing) giúp API giao tiếp với frontend hoặc các ứng dụng khác.



*Hình 16: Khởi tạo Flask App*

**1.3 Tương tác và Đo lường Hiệu suất:**

Interactive Environment (Môi trường Tương tác): sử dụng môi trường tương tác Jupyter Notebook, Google Colab để xây dựng, huấn luyện, và đánh giá các mô hình học máy một cách tương tác, giao diện đồ họa cho phép chạy từng dòng lệnh để xem kết quả ngay lập tức.

Performance Measurement (Đo lường Hiệu suất): Tính toán các chỉ số đánh giá hiệu suất như độ chính xác, độ phân loại, đồng nhất, và F1-score để đánh giá hiệu suất của mô hình.

Real-Time Model Training (Huấn luyện mô hình theo thời gian thực): Cho phép quan sát quá trình huấn luyện mô hình theo thời gian thực,hiển thị loss function, accuracy, hoặc các metric khác sau từng epoch để theo dõi quá trình tối ưu mô hình.

**1.4 Trực quan Hóa :**

Data Visualization (Trực quan Hóa Dữ liệu): Hỗ trợ trực quan hóa dữ liệu để hiểu rõ hơn về đặc điểm và phân bố của dữ liệu. Sử dụng Seaborn (sns.histplot()) để vẽ biểu đồ phân phối của các cột dữ liệu trước và sau khi xử lý KNN Imputer.

Performance Charts (Biểu đồ Hiệu suất): Tạo biểu đồ đường và biểu đồ phân phối để phân tích hiệu suất của mô hình và so sánh giữa các mô hình khác nhau.

Confusion Matrix (Ma trận Confusion): Trực quan hóa ma trận confusion để đánh giá hiệu suất phân loại của mô hình trên dữ liệu kiểm thử.

**1.5 Công cụ Phân tích Dữ liệu:**

Data Processing (Xử lý Dữ liệu): Tích hợp tốt với các công cụ phân tích dữ liệu như Pandas và NumPy để xử lý và làm sạch dữ liệu trước khi huấn luyện mô hình.

External Data Integration (Tích hợp với Dữ liệu Ngoại vi): Hỗ trợ đọc và xử lý dữ liệu từ nhiều nguồn khác nhau như tệp CSV và API.

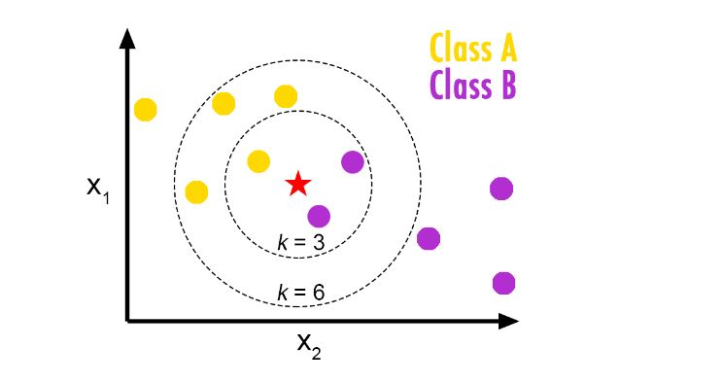
Feature Engineering (Kỹ thuật Xây dựng Đặc trưng): Gồm các phương pháp chuyển đổi, tạo mới hoặc chọn lọc các đặc trưng (features) quan trọng để cải thiện hiệu suất mô hình. Xử lý dữ liệu thiếu bằng KNN Imputer giúp cải thiện dữ liệu đầu vào cho mô hình

Exploratory Data Analysis (Phân Tích Dữ Liệu Thăm Dò - EDA): Dùng để khám phá dữ liệu bằng cách hiển thị thông tin tổng quát, thống kê, trực quan hóa để hiểu về dữ liệu trước khi huấn luyện mô hình. So sánh dữ liệu trước và sau khi xử lý bằng biểu đồ.

Statistical Analysis (Phân Tích Thống Kê): Sử dụng các phương pháp thống kê như trung bình, phương sai, độ lệch chuẩn để đánh giá dữ liệu.

1. **Phương pháp**

**2.1 Tổng quan về kNN**



*Hình 17:Giải thuật kNN*

Trong lĩnh vực máy học, các thuật toán có thể được chia thành hai nhóm chính: có giám sát (supervised learning) và không giám sát (unsupervised learning). Các thuật toán có giám sát yêu cầu dữ liệu đầu vào phải được gán nhãn, giúp mô hình học cách dự đoán hoặc phân loại dựa trên thông tin đã biết. Ngược lại, các thuật toán không giám sát không yêu cầu dữ liệu có nhãn, thay vào đó, chúng tìm kiếm mẫu hoặc cấu trúc ẩn trong dữ liệu, chẳng hạn như nhóm cụm (clustering) hoặc giảm chiều dữ liệu bằng các phương pháp như phân tích thành phần chính (PCA).

Khác với học không giám sát, các kỹ thuật có giám sát yêu cầu dữ liệu đầu vào phải được gắn nhãn và thường được chia thành hai tập: tập huấn luyện và tập kiểm thử. Những thuật toán này chủ yếu được sử dụng cho các bài toán phân loại (chẳng hạn như nhận diện chữ số viết tay) hoặc hồi quy (như dự đoán giá trị của một căn nhà dựa trên diện tích và vị trí). Trong các thuật toán phân loại có giám sát, mô hình chỉ có thể xác định nhãn của một đối tượng nếu nhãn đó đã xuất hiện trong tập huấn luyện. Việc chia dữ liệu thành hai phần giúp đánh giá mô hình trên dữ liệu mới chưa từng thấy trước đó. Sau khi huấn luyện, hiệu suất của mô hình được kiểm tra thông qua các thước đo như ma trận nhầm lẫn (Confusion Matrix) và các chỉ số đánh giá như độ chính xác, độ nhạy và độ đặc hiệu.

Một thuật toán phân loại phổ biến thuộc nhóm có giám sát là k-Nearest Neighbors (kNN). Đây là một thuật toán đơn giản nhưng hiệu quả, dựa trên nguyên lý rằng một điểm dữ liệu mới sẽ thuộc vào nhóm của k điểm lân cận gần nhất của nó. kNN có thể được sử dụng cho cả bài toán phân loại và hồi quy. Trong bài toán phân loại, nhãn của điểm dữ liệu mới được xác định bằng cách lấy nhãn phổ biến nhất trong k điểm lân cận. Trong bài toán hồi quy, giá trị của điểm mới được tính bằng cách trung bình giá trị của k điểm lân cận gần nhất.

Thuật toán kNN có nhiều ứng dụng thực tế, bao gồm nhận dạng chữ viết tay, phát hiện gian lận, phân loại bệnh tật và đề xuất sản phẩm. Trong lĩnh vực y tế, kNN có thể được sử dụng để dự đoán một bệnh nhân có mắc bệnh hay không dựa trên dữ liệu của các bệnh nhân trước đó. Một cơ sở dữ liệu bệnh nhân được chia thành tập huấn luyện để huấn luyện mô hình và tập kiểm tra để đánh giá. Khi có dữ liệu của một bệnh nhân mới, mô hình sẽ dựa trên khoảng cách giữa bệnh nhân đó với các bệnh nhân trong tập huấn luyện để xác định kết quả.

Thuật toán kNN yêu cầu một số giả định và điều kiện để hoạt động hiệu quả:

+ Dữ liệu đầu vào cần được chuẩn hóa để tránh ảnh hưởng của các thang đo khác nhau.

+ Khoảng cách giữa các điểm dữ liệu thường được tính bằng khoảng cách Euclidean, Manhattan hoặc Minkowski.

+ Giá trị của k phải được chọn một cách phù hợp để cân bằng giữa độ phức tạp và độ chính xác của mô hình.

+ Dữ liệu huấn luyện nên có đủ điểm để mô hình có thể tổng quát tốt.

**2.2 Lý do lựa chọn kNN cho bài toán**

Một trong những lý do chính để chọn giải thuật kNN cho đề tài này là do tính chất đơn giản nhưng hiệu quả của thuật toán trong việc xử lý bài toán phân loại bệnh tiểu đường. Bộ dữ liệu 'diabetes.arff' chứa các đặc trưng y tế liên quan đến bệnh tiểu đường, như nồng độ glucose trong máu, huyết áp, chỉ số BMI, và số lần mang thai. Các đặc trưng này có thể được sử dụng để dự đoán nguy cơ mắc bệnh của một bệnh nhân mới, biến bài toán thành một bài toán phân loại điển hình.

Thuật toán kNN phù hợp với bài toán này vì nó không yêu cầu giả định mạnh mẽ về phân phối của dữ liệu, điều này rất hữu ích trong bối cảnh dữ liệu y tế thường không tuân theo một phân phối chuẩn. Hơn nữa, kNN có thể dễ dàng thích ứng với các đặc trưng khác nhau bằng cách điều chỉnh giá trị k và lựa chọn thước đo khoảng cách phù hợp, chẳng hạn như khoảng cách Euclidean hoặc Manhattan. Điều này giúp kNN có khả năng nhận diện tốt các mẫu dữ liệu tương đồng để đưa ra dự đoán chính xác.

Ngoài ra, kNN hoạt động tốt với dữ liệu có số lượng mẫu không quá lớn, điều này phù hợp với tập dữ liệu 'diabetes.arff' vì nó có kích thước vừa phải. Việc triển khai kNN cũng tương đối dễ dàng, giúp đơn giản hóa quá trình xây dựng mô hình và tập trung vào việc phân tích, đánh giá kết quả.

Với những ưu điểm trên, kNN là một lựa chọn hợp lý để khai phá dữ liệu 'diabetes.arff', giúp xác định các yếu tố quan trọng ảnh hưởng đến nguy cơ mắc bệnh tiểu đường, đồng thời cung cấp một phương pháp dự đoán trực quan và hiệu quả.

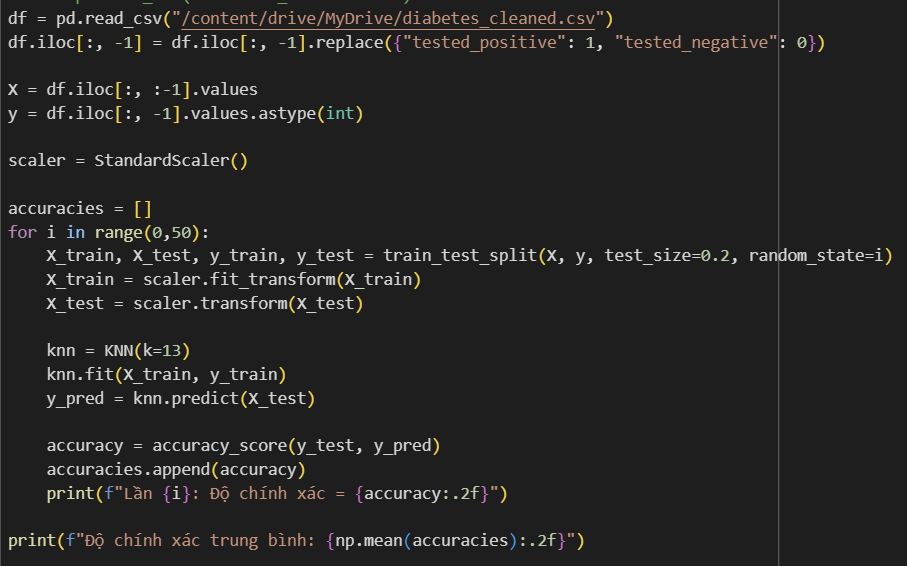
Mặc dù kNN có nhiều ưu điểm như đơn giản, dễ triển khai, và có thể áp dụng cho nhiều bài toán khác nhau, nhưng nó cũng có một số hạn chế. Vì là một thuật toán dựa trên khoảng cách, kNN có thể bị ảnh hưởng bởi dữ liệu nhiễu và không hiệu quả khi số lượng đặc trưng quá lớn. Ngoài ra, chi phí tính toán có thể cao khi kích thước dữ liệu lớn do cần tính toán khoảng cách cho từng điểm mới. Để cải thiện hiệu suất của kNN, có thể sử dụng các phương pháp như giảm chiều dữ liệu (PCA), chọn lựa đặc trưng quan trọng, hoặc sử dụng cấu trúc dữ liệu chuyên biệt như KD-Tree và Ball-Tree để tăng tốc quá trình tìm kiếm lân cận.

**2.3 Triển khai giải thuật và huấn luyện mô hình**

Tiến hành triển khai thuật toán K-Nearest Neighbors (KNN) để dự đoán bệnh tiểu đường dựa trên tập dữ liệu diabetes\_cleaned.csv. Mục tiêu chính là kiểm tra hiệu suất mô hình trước và sau khi chuẩn hóa dữ liệu, đồng thời đánh giá sự thay đổi các chỉ số đo lường hiệu suất như **Accuracy**, **Precision**, **Recall**, **F1-score**.

**2.3.1 Đánh giá độ ổn định của mô hình kNN:**

Dánh giá độ ổn định của mô hình kNN bằng cách kiểm tra độ chính xác của mô hình trên 50 lần chia dữ liệu ngẫu nhiên khác nhau (tương ứng với random\_state chạy từ 0 đến 49).



Hình 18: Kiểm tra độ chính xác của mô hình trên **50 lần chia dữ liệu ngẫu nhiên khác nhau**

Định nghĩa lớp kNN:

Hàm \_\_**init**\_\_: Khởi tạo đối tượng KNN với số lượng láng giềng gần nhất (k).

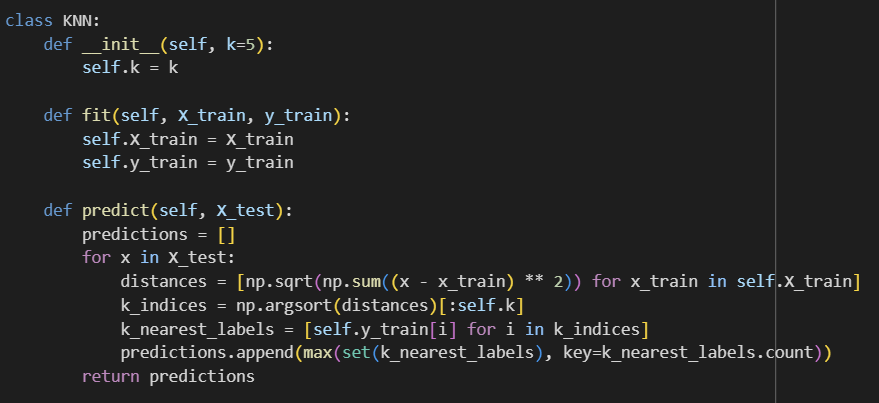
Hàm **fit**: Lưu dữ liệu huấn luyện (X\_train, y\_train) để dùng trong dự đoán.

Hàm **predict**:

+ Tính khoảng cách Euclidean từ điểm dữ liệu mới đến tất cả các điểm trong tập train.

+ Sắp xếp các khoảng cách và lấy k điểm gần nhất.

+ Lấy nhãn xuất hiện nhiều nhất trong k điểm gần nhất để dự đoán.



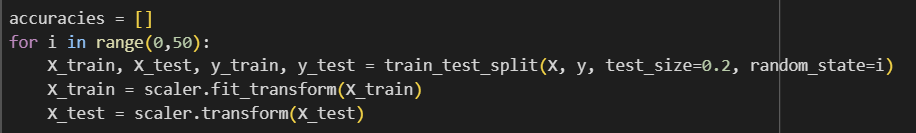
*Hình 19: Định nghĩa lớp kNN*

Tính độ chính xác với các random\_state khác nhau:

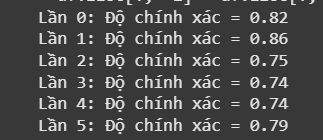
Chạy vòng lặp 50 lần với các giá trị random\_state khác nhau (i chạy từ 0 → 49)

Chia tập dữ liệu với tỷ lệ 80% train, 20% test.

Chuẩn hóa dữ liệu train/test với StandardScaler.



*Hình 20: Vòng lặp tính độ chính xác với các random\_state khác nhau*



*Hình 21: random\_state =1 có độ chính xác cao nhất*

Mục đích chính:

Kiểm tra sự ảnh hưởng của cách chia dữ liệu train/test đến hiệu suất mô hình.

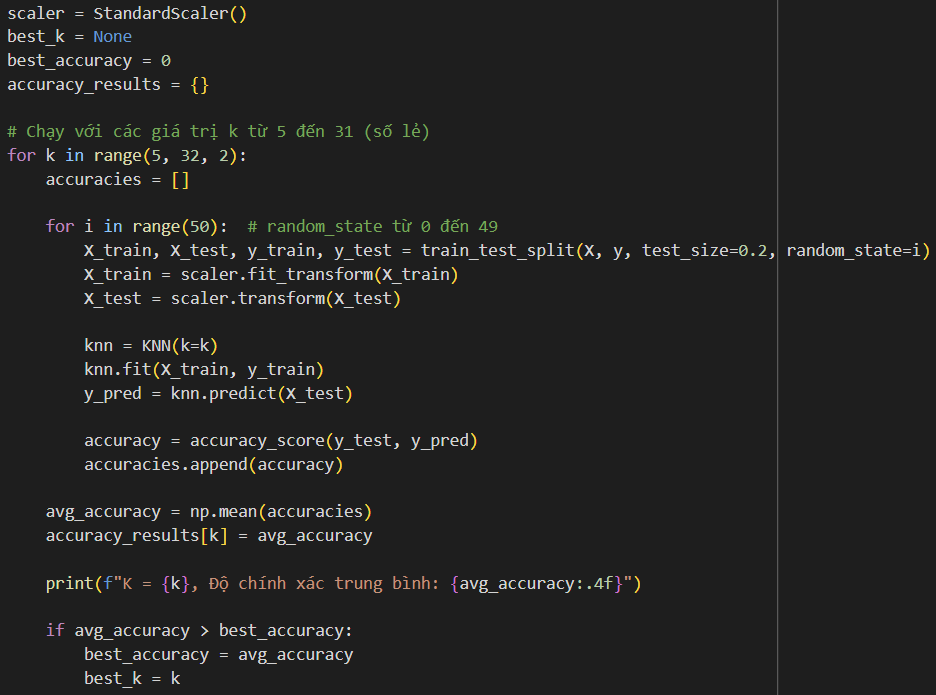
+ Khi thay đổi random\_state, dữ liệu được chia lại theo cách khác nhau, ảnh hưởng đến việc huấn luyện và kiểm tra mô hình.

+ Việc chạy nhiều lần giúp đánh giá độ ổn định của mô hình KNN.

Tính độ chính xác trung bình của mô hình.

+ Thay vì chỉ chạy một lần với một giá trị random\_state cố định, chúng ta chạy 50 lần và lấy trung bình để có cái nhìn tổng quan hơn về hiệu suất mô hình.

### 2.3.2 Tìm giá trị k tối ưu cho mô hình kNN với độ chính xác trung bình :



***Hình 22: Tìm kiếm giá trị k tốt nhất***

Lặp qua các giá trị k từ 5 đến 31 (chỉ số lẻ) để kiểm tra thuật toán.

Với mỗi giá trị k, chạy thử nghiệm 50 lần với các random\_state khác nhau.

Ở mỗi lần thử nghiệm:

+ Chia dữ liệu thành tập train và test (test\_size=0.2).

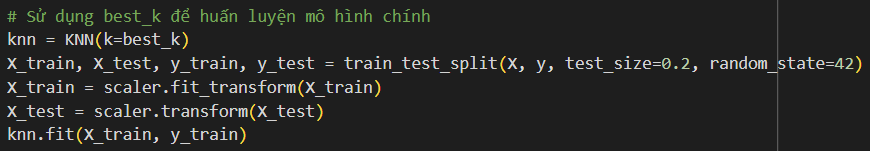
+ Chuẩn hóa dữ liệu bằng StandardScaler (quan trọng với KNN để tránh khoảng cách bị lệch).

+ Huấn luyện mô hình KNN với k hiện tại.

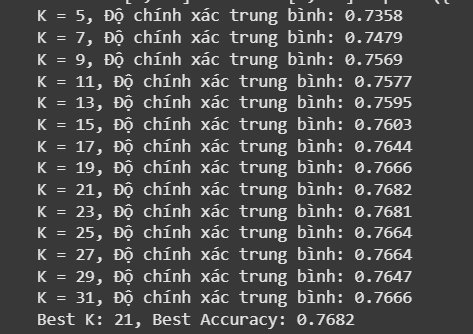
+ Dự đoán trên tập test và tính độ chính xác.

Lưu độ chính xác trung bình của 50 lần chạy vào accuracy\_results[k].

Cập nhật best\_k nếu avg\_accuracy của k hiện tại cao hơn best\_accuracy.

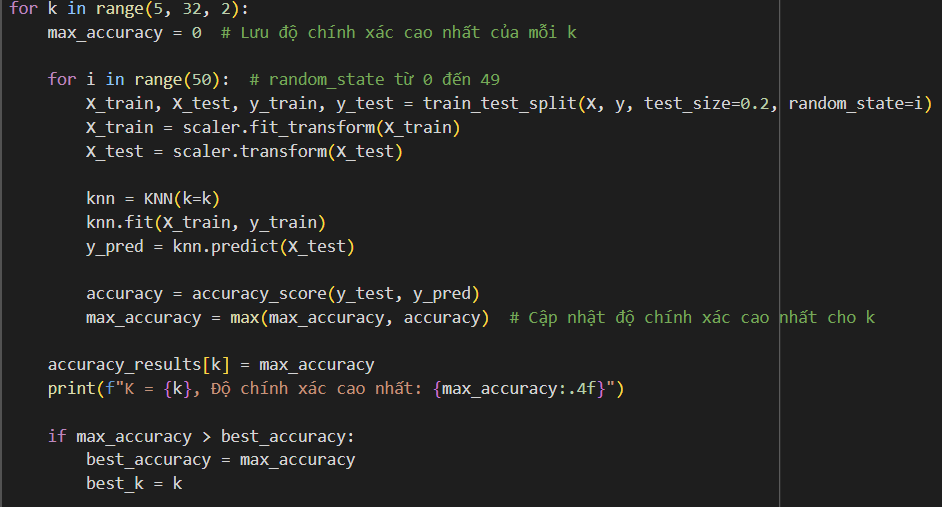


*Hinh 23: Sử dụng best k để huấn luyện mô hình chính*



*Hình 24: Kết quả sau khi chạy*

### 2.3.3 Tìm giá trị k tối ưu cho mô hình kNN với độ chính xác cao nhất :



*Hình 25 :Chạy thử nghiệm với các giá trị k từ 5 đến 31 (chỉ số lẻ)*

Lặp qua các giá trị k từ 5 đến 31 (chỉ số lẻ) để kiểm tra thuật toán.

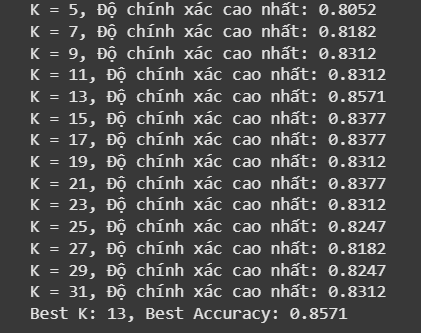
Chạy 50 lần với các random\_state khác nhau để đảm bảo tính ổn định của mô hình.

Chuẩn hóa dữ liệu train/test để giảm ảnh hưởng của giá trị đặc trưng quá lớn hoặc quá nhỏ.

Lưu độ chính xác cao nhất của mỗi k.

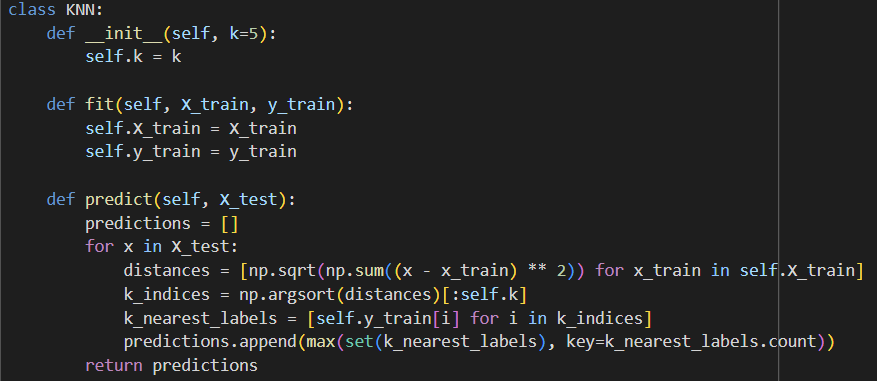
Huấn luyện mô hình KNN với best\_k được tìm thấy.

Chuẩn hóa dữ liệu train/test.



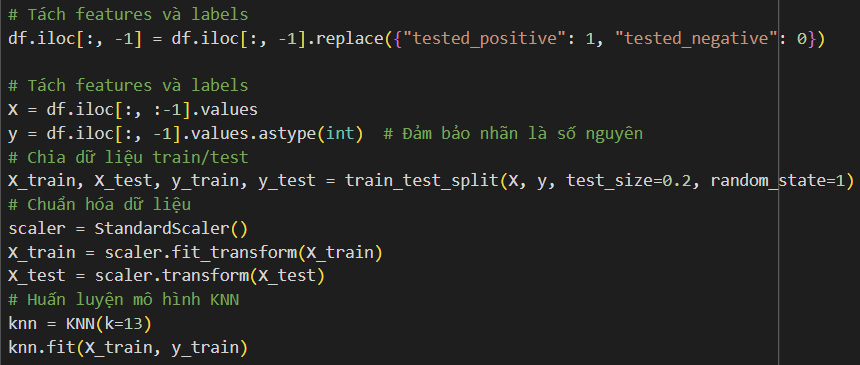
*Hình 26: Best K được tìm thấy là 13, Best Accuracy :0.8571*

**2.3.4 Xây dựng thuật toán KNN .**



*Hình 27 : Định nghĩa thuật toán kNN*

**Tiền xử lý dữ liệu**



*Hình 28: Xử lý dữ liệu*

Tách dữ liệu thành features và labels

Chia tập huấn luyện và kiểm tra(chia dữ liệu thành tập huấn luyện (80%) và tập kiểm tra (20%)).

Chuẩn hóa dữ liệu:

Dùng **StandardScaler** để chuẩn hóa dữ liệu (đưa về phân phối chuẩn có trung bình = 0, độ lệch chuẩn = 1).

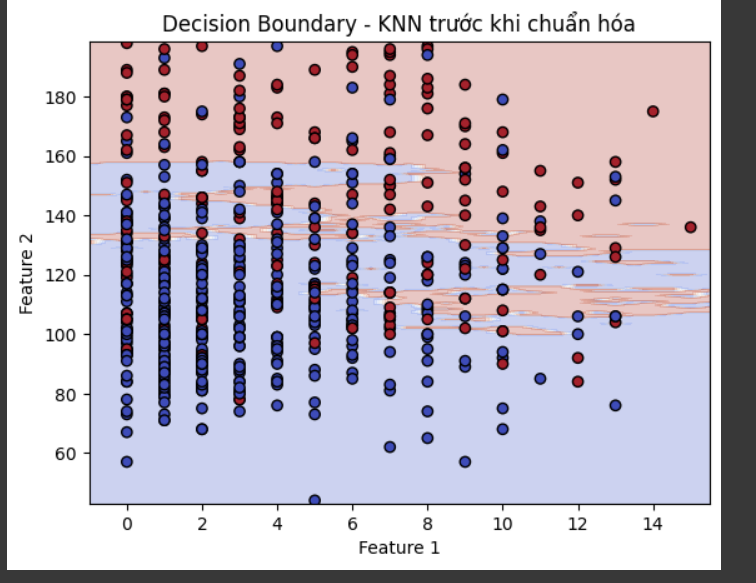
**fit\_transform(X\_train)**: Tính toán và áp dụng chuẩn hóa trên tập huấn luyện.

**transform(X\_test)**: Áp dụng chuẩn hóa trên tập kiểm tra dựa trên các giá trị tính toán từ tập huấn luyện.

Huấn luyện mô hình KNN với k =13.

# CHƯƠNG IV: ĐÁNH GIÁ MÔ HÌNH

1. **Mô hình kNN trước khi chuẩn hóa.**



*Hình 29: Decision Boundary - KNN trước khi chuẩn hóa*

Hình dạng của đường ranh giới (Decision Boundary)

+ Vì KNN là một thuật toán dựa trên khoảng cách, đường ranh giới sẽ không phải là một đường thẳng đơn giản như mô hình tuyến tính mà sẽ có dạng cong hoặc phân đoạn.

+ Trước khi chuẩn hóa, các đặc trưng có thể có đơn vị đo lường khác nhau, khiến khoảng cách giữa các điểm bị ảnh hưởng không đồng đều. Điều này có thể làm cho ranh giới quyết định trở nên không mượt mà, có nhiều phân đoạn và chưa tối ưu.

Ảnh hưởng của dữ liệu chưa chuẩn hóa

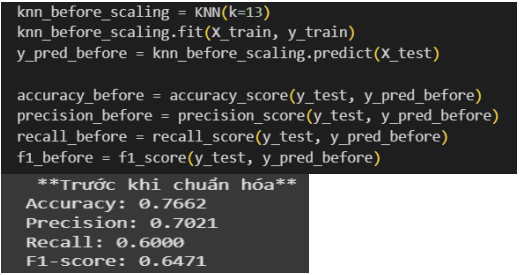
+ KNN bị ảnh hưởng bởi đặc trưng có phạm vi lớn hơn.

+ Mô hình ưu tiên một số đặc trưng hơn những đặc trưng khác, làm giảm hiệu suất phân loại.

Dự đoán vùng biên kém chính xác

+ Nếu mô hình tạo ra nhiều vùng biên phức tạp với nhiều đường cong gấp khúc, điều đó có thể phản ánh việc mô hình bị ảnh hưởng bởi sự phân bố không đồng đều của dữ liệu.

+ False Positives (dự đoán nhầm người không mắc bệnh thành mắc bệnh) và False Negatives (bỏ sót bệnh nhân mắc bệnh) có thể cao hơn do ranh giới phân tách chưa hợp lý.



*Hình 30: Mô hình trước khi chuẩn hóa*

Hiệu suất tổng thể ở mức trung bình (Accuracy ≈ 76.62%),Mức độ chính xác khá ổn, nhưng vẫn có nhiều sai số (~23.38% mẫu dự đoán sai).

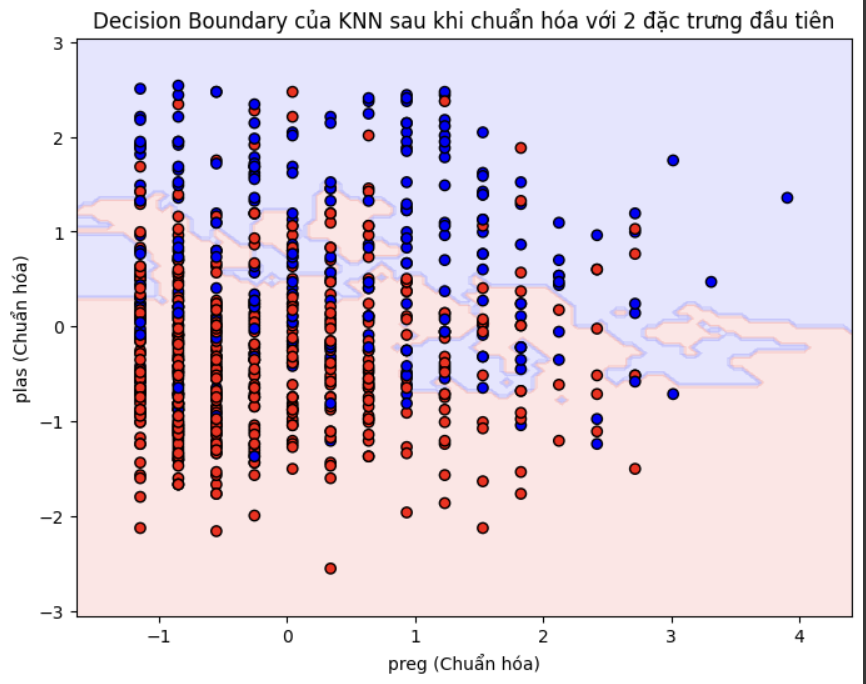
Precision khá tốt (70.21%), nhưng Recall thấp (60.00%), nghĩa là mô hình bỏ sót nhiều bệnh nhân bị tiểu đường thực sự,Precision không quá cao, có nghĩa là mô hình vẫn dự đoán nhầm một số người khỏe mạnh thành bị tiểu đường (False Positive cao).

Recall = 60.00% giá trị này thấp, tức là mô hình bỏ sót đến 40% bệnh nhân tiểu đường thực sự (False Negative cao).

F1-score (64.71%) không cao, chứng tỏ mô hình chưa tối ưu trong việc cân bằng Precision và Recall,Mức F1-score trung bình, thể hiện rằng mô hình vẫn cần cải thiện, đặc biệt là về Recall.

KNN có thể gặp vấn đề với dữ liệu không được chuẩn hóa, vì các đặc trưng có thể có đơn vị đo khác nhau, khiến khoảng cách bị méo mó.

1. **Mô hình kNN sau khi chuẩn hóa.**



*Hình 31: Decison Boundary sau khi chuẩn hóa*

Sau khi chuẩn hóa dữ liệu, mô hình KNN có sự cải thiện rõ rệt, thể hiện qua các đặc điểm sau:

Đường ranh giới mượt mà hơn

+ Trước khi chuẩn hóa, ranh giới có thể bị méo mó do ảnh hưởng của các đặc trưng có thang đo khác nhau.

+ Sau khi chuẩn hóa, đường ranh giới trở nên mượt mà, cân đối hơn vì khoảng cách giữa các điểm dữ liệu được tính toán đồng đều.

Phân tách hai lớp dữ liệu rõ ràng hơn

+ Sau chuẩn hóa, các điểm dữ liệu được sắp xếp hợp lý, giúp mô hình phân biệt tốt hơn giữa hai lớp "tested\_positive" (bị tiểu đường) và "tested\_negative" (không bị tiểu đường).

+ Điều này góp phần giảm số lượng dự đoán sai (False Positive và False Negative).

Hiệu suất mô hình cải thiện

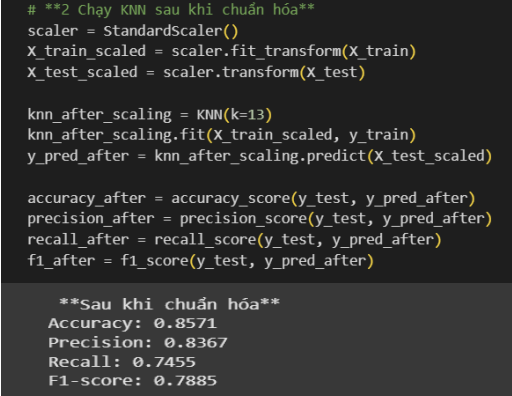
+ Từ kết quả, ta thấy rằng Accuracy đã tăng từ 76.62% lên 85.71%, cho thấy mô hình hoạt động hiệu quả hơn sau chuẩn hóa.

+ Precision và Recall cũng tăng đáng kể, chứng tỏ mô hình ít bỏ sót bệnh nhân tiểu đường hơn và cũng ít dự đoán nhầm người không mắc bệnh thành mắc bệnh.

Mô hình ổn định hơn

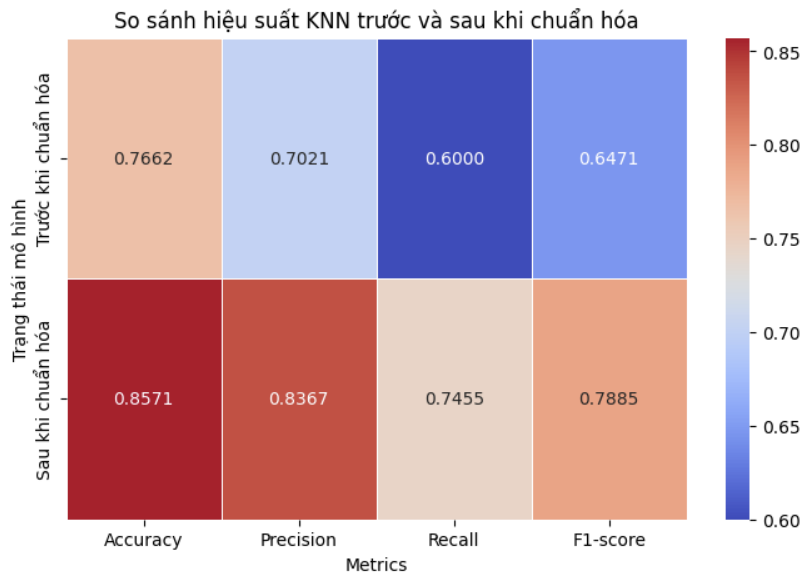
+ Khi các đặc trưng được chuẩn hóa về cùng một thang đo, thuật toán KNN ít bị ảnh hưởng bởi các thuộc tính có giá trị lớn hơn.

+ Điều này giúp đảm bảo rằng mô hình hoạt động nhất quán hơn trên toàn bộ dữ liệu.



*Hình 32: Mô hình sau khi chuẩn hóa*

1. **Nhận xét mô hình kNN.**



*Hình 33: So sánh hiệu suất mô hình kNN*

Hiệu suất tổng thể được cải thiện đáng kể

Accuracy tăng từ 76.62% → 85.71% (+9.09%)

Điều này chứng tỏ rằng chuẩn hóa dữ liệu giúp mô hình KNN hoạt động tốt hơn, giảm sai số đáng kể.

Precision tăng mạnh (+13.46%)

Trước chuẩn hóa, Precision chỉ đạt 70.21%, có nghĩa là mô hình dự đoán sai khá nhiều người không bị tiểu đường thành bị tiểu đường (False Positive cao).

Sau chuẩn hóa, Precision tăng lên 83.67%, tức là ít dự đoán nhầm người không bị tiểu đường thành có bệnh hơn.

Giá trị cao hơn → Ít cảnh báo giả (False Positive giảm).

Recall tăng mạnh nhất (+14.55%)

Trước chuẩn hóa, Recall chỉ 60%, tức là 40% bệnh nhân tiểu đường thực sự bị bỏ sót (False Negative cao).

Sau chuẩn hóa, Recall tăng lên 74.55%, có nghĩa là mô hình phát hiện đúng bệnh nhân tiểu đường nhiều hơn.

Điều này cực kỳ quan trọng, vì bỏ sót bệnh nhân tiểu đường có thể gây hậu quả nghiêm trọng.

F1-score cũng tăng đáng kể (+14.14%)

F1-score tăng từ 64.71% → 78.85%, chứng tỏ mô hình cân bằng Precision & Recall tốt hơn.

Mô hình trở nên ổn định hơn sau khi chuẩn hóa.

1. **Kết luận.**

Sau khi đánh giá hiệu suất mô hình,chúng ta rút ra được:

+ KNN đơn giản, dễ triển khai: Không cần quá nhiều tài nguyên tính toán như các mô hình phức tạp khác (SVM, Neural Networks).

+ KNN hoạt động tốt với dữ liệu chuẩn hóa: Kết quả thử nghiệm cho thấy chuẩn hóa giúp KNN cải thiện đáng kể.

+ Dữ liệu "diabetes\_cleaned.csv" không quá lớn: KNN phù hợp vì nó hoạt động hiệu quả với các tập dữ liệu vừa phải.

+ Dễ dàng tích hợp API: KNN có thể được triển khai nhanh chóng vào ứng dụng web hoặc mobile, giúp người dùng nhận kết quả dự đoán.

Dựa vào kết quả trên, mô hình KNN sau khi chuẩn hóa có độ chính xác và ổn định cao , đủ tin cậy để triển khai trong một ứng dụng thực tế.Với hiệu suất đã được cải thiện, mô hình KNN đạt yêu cầu triển khai vào một ứng dụng giúp người dùng kiểm tra nguy cơ tiểu đường thông qua các triệu chứng.

# CHƯƠNG V: XÂY DỰNG ỨNG DỤNG

1. **Giới thiệu về Flask và React**

**Flask**

Flask là một framework nhẹ của Python dùng để xây dựng các ứng dụng web, API RESTful một cách đơn giản và linh hoạt. Flask hỗ trợ:

+ Xử lý request/response HTTP

+ Tích hợp với các thư viện AI/ML (như scikit-learn, NumPy, Pandas)

+ CORS để cho phép client (React) gọi API từ một nguồn khác

**React**

React là một thư viện JavaScript mạnh mẽ dùng để xây dựng giao diện người dùng (UI). Nó giúp tạo các component tái sử dụng, có trạng thái riêng (state) và có thể cập nhật giao diện một cách hiệu quả.

1. **Code chương trình**

**2.1 Backend - Flask (Xử lý dữ liệu và API dự đoán)**

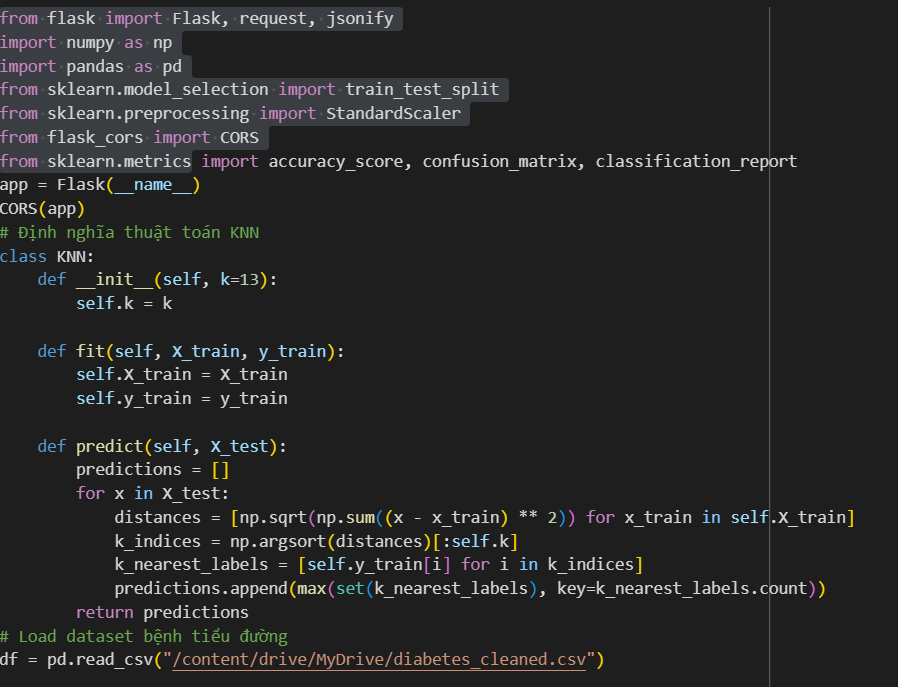
**Chức năng chính của code**

Tải dữ liệu tiểu đường (diabetes\_cleaned.csv)

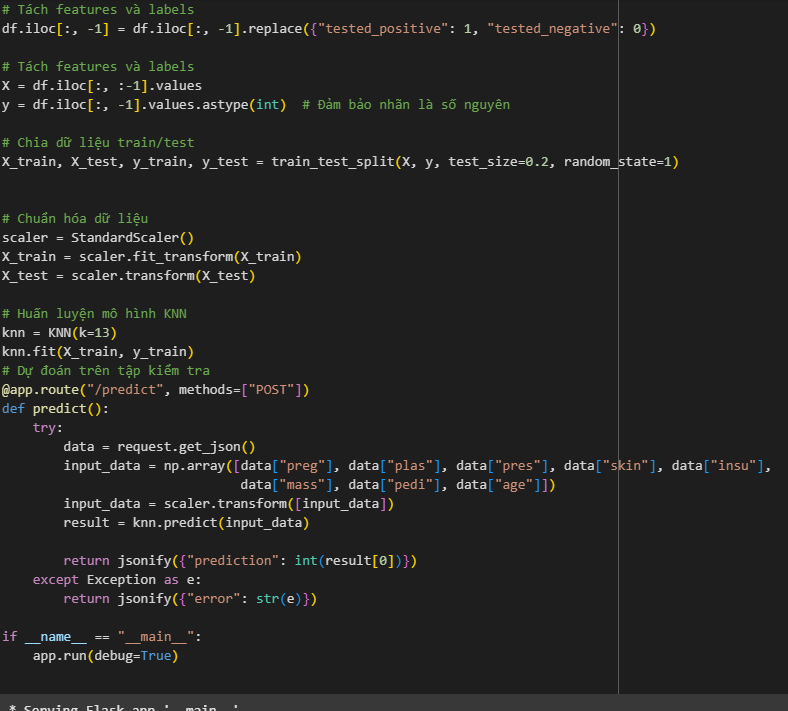
Huấn luyện mô hình KNN (K-Nearest Neighbors)

Chuẩn hóa dữ liệu bằng StandardScaler

Triển khai API /predict nhận dữ liệu từ frontend, xử lý và trả kết quả



*Hình 34: Định nghĩa thuật toán kNN*



*Hình 35: Xây dựng API*

API dự đoán /predict:

Nhận dữ liệu đầu vào từ React

Tiền xử lý dữ liệu đầu vào (chuẩn hóa)

Dự đoán bằng KNN

Trả về kết quả JSON { "prediction": 0 hoặc 1 }

Nếu trả về =0 tức là âm tính

Nếu trả về=1 tức là dương tính

**2.2 Frontend - React (Giao diện nhập dữ liệu & hiển thị kết quả)**

**Chức năng chính của React**

Hiển thị form nhập dữ liệu

Gửi dữ liệu đến API Flask

Hiển thị kết quả dự đoán (Âm tính hoặc Dương tính)

**Code chương trình:**

import { useState } from "react";

export default function KnnInputForm() {

  const [formData, setFormData] = useState({

    preg: "",

    plas: "",

    pres: "",

    skin: "",

    insu: "",

    mass: "",

    pedi: "",

    age: "",

  });

  const [errors, setErrors] = useState({});

  const [result, setResult] = useState(null);

  const [loading, setLoading] = useState(false);

  const validate = () => {

    let newErrors = {};

    if (formData.preg < 0) newErrors.preg = "Số lần mang thai phải >= 0";

    ["plas", "pres", "skin", "insu", "mass", "pedi", "age"].forEach((key) => {

      if (formData[key] <= 0) newErrors[key] = "Giá trị phải > 0";

    });

    setErrors(newErrors);

    return Object.keys(newErrors).length === 0;

  };

  const handleChange = (e) => {

    setFormData({ ...formData, [e.target.name]: e.target.value });

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    if (!validate()) return;

    setLoading(true);

    setResult(null);

    try {

      const response = await fetch("http://127.0.0.1:5000/predict", {

        method: "POST",

        headers: { "Content-Type": "application/json" },

        body: JSON.stringify({

          preg: parseFloat(formData.preg),

          plas: parseFloat(formData.plas),

          pres: parseFloat(formData.pres),

          skin: parseFloat(formData.skin),

          insu: parseFloat(formData.insu),

          mass: parseFloat(formData.mass),

          pedi: parseFloat(formData.pedi),

          age: parseFloat(formData.age),

        }),

      });

      const data = await response.json();

      if (data.error) {

        setResult("Lỗi: " + data.error);

      } else {

        setResult(data.prediction === 1 ? "🩺 Dương tính (Có nguy cơ mắc tiểu đường)" : " Âm tính (Không mắc tiểu đường)");

      }

    } catch (error) {

      setResult("Lỗi kết nối API");

    } finally {

      setLoading(false);

    }

  };

  return (

    <div className="flex justify-center items-center min-h-screen bg-gray-100">

      <div className="w-full max-w-lg p-8 bg-white shadow-lg rounded-2xl">

        <h2 className="text-2xl font-bold text-center mb-6 text-gray-800">🩺 Dự Đoán Bệnh Tiểu Đường</h2>

        <form onSubmit={handleSubmit}>

          {Object.keys(formData).map((key) => (

            <div key={key} className="mb-4">

              <label className="block text-gray-700 font-medium capitalize">{key}:</label>

              <input

                type="number"

                name={key}

                value={formData[key]}

                onChange={handleChange}

                className="w-full p-3 border border-gray-300 rounded-lg mt-1 focus:outline-none focus:ring-2 focus:ring-blue-500"

                placeholder={`Nhập ${key}`}

              />

              {errors[key] && <p className="text-red-500 text-sm mt-1">{errors[key]}</p>}

            </div>

          ))}

          <button

            type="submit"

            className="w-full bg-blue-600 text-white py-3 rounded-lg font-semibold text-lg mt-4 hover:bg-blue-700 transition duration-200"

            disabled={loading}

          >

            {loading ? "Đang dự đoán..." : ""}

          </button>

        </form>

        {result && (

          <p className={`text-center text-lg font-semibold mt-4 ${result.includes("Dương tính") ? "text-red-600" : "text-green-600"}`}>

            {result}

          </p>

        )}

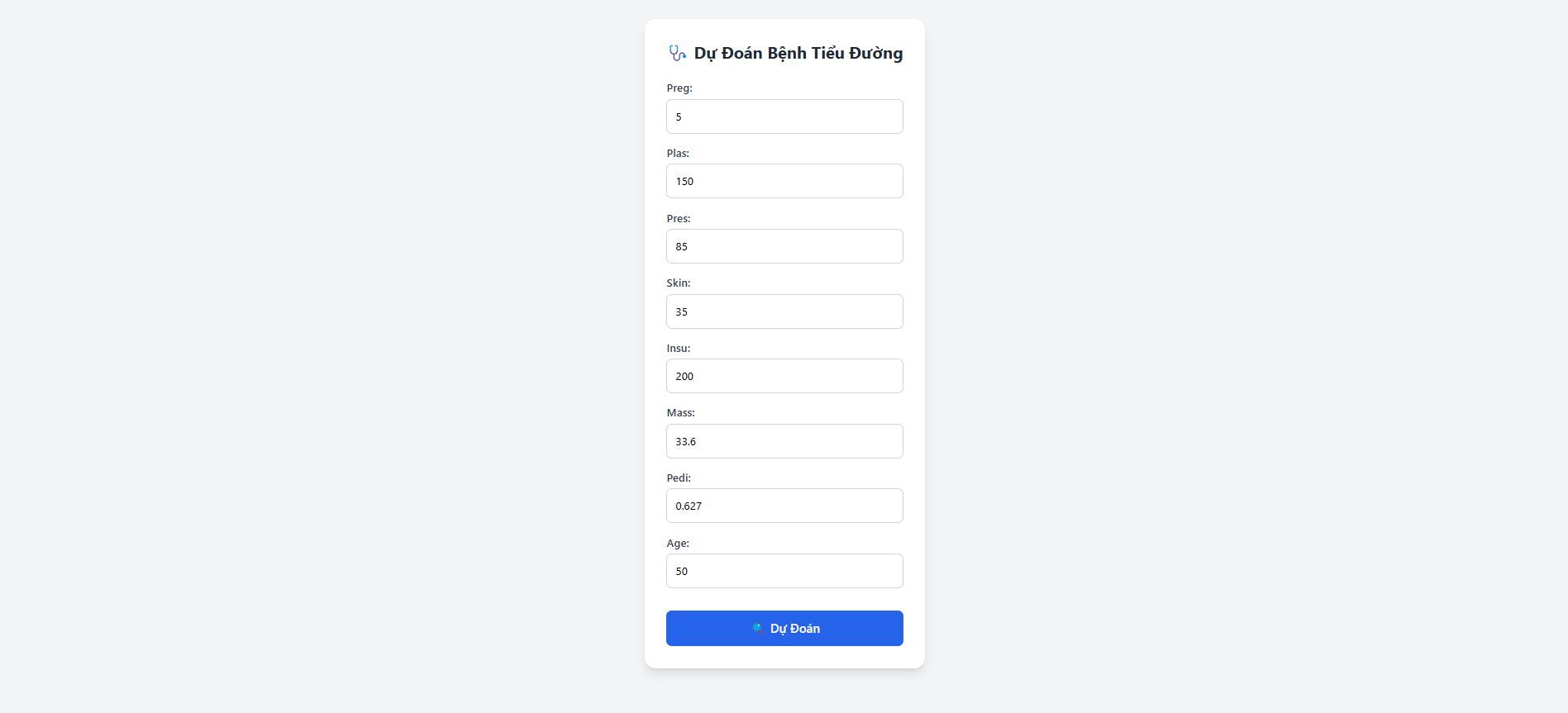
      </div>

    </div>

  );

}

**2.3 Demo chương trình**

****

*Hình 36: Giao diện*

Giao diện bao gồm 8 trường nhập liệu (input field) để người dùng nhập thông tin cần thiết:

Preg - Nhập số lần mang thai

Plas - Nhập mức đường huyết

Pres - Nhập huyết áp

Skin - Nhập độ dày nếp gấp da

Insu - Nhập mức insulin

Mass - Nhập chỉ số BMI

Pedi - Nhập chỉ số di truyền

Age - Nhập tuổi

Phía dưới biểu mẫu có một nút bấm màu xanh "Dự Đoán" giúp người dùng gửi thông tin để thực hiện dự đoán bệnh tiểu đường.

Thông tin dự đoán sẽ được hiện ngay dưới nút màu xanh

Khi người dùng bấm nút Dự đoán:

Ở FrondEnd

Kiểm tra dữ liệu đầu vào xem có hợp lệ không.

Gửi yêu cầu POST đến API Flask tại [/predict](http://127.0.0.1:5000/predict)

Dữ liệu gửi đi là JSON, chứa các thông số sức khỏe người dùng nhập.

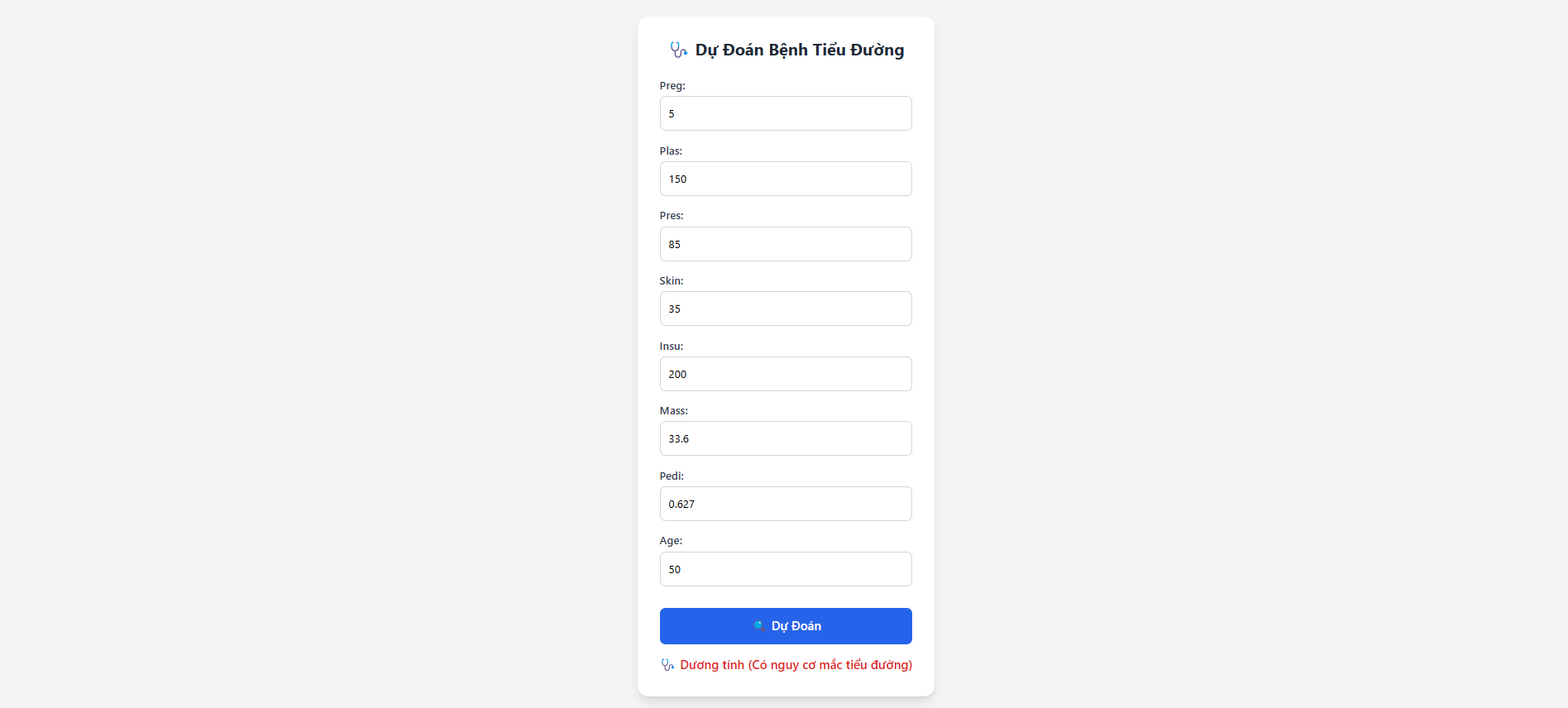
ỏ phía Backend:

Flask nhận request từ React qua API /predict

Trích xuất dữ liệu từ JSON và chuyển thành mảng numpy

Dùng mô hình KNN đã huấn luyện (knn\_model.pkl) để dự đoán

Trả kết quả JSON ({"prediction": 0} hoặc {"prediction": 1}) về frontend.



*Hình 37: Chương trình sẽ in ra kết quả dự đoán có thể bị tiểu đường hay không.*

# 

# CHƯƠNG VI: KẾT LUẬN

Bài tiểu luận đã thực hiện khai phá bộ dữ liệu diabetes.arff để xây dựng công cụ hỗ trợ chẩn đoán nguy cơ mắc bệnh tiểu đường. Quá trình nghiên cứu đã trải qua các bước quan trọng, bao gồm đặc tả dữ liệu, lựa chọn thuật toán khai phá phù hợp, đánh giá hiệu suất mô hình trước và sau khi chuẩn hóa, và cuối cùng là xây dựng một ứng dụng giúp người dùng nhập triệu chứng để dự đoán nguy cơ mắc bệnh.

Kết quả cho thấy, việc sử dụng thuật toán k-Nearest Neighbors (kNN) với số lượng láng giềng tối ưu k = 13 đã mang lại độ chính xác cao trong việc phân loại bệnh nhân tiểu đường. Đặc biệt, sau khi chuẩn hóa dữ liệu, mô hình đã có sự cải thiện rõ rệt về Accuracy (+9.09%), Precision (+13.46%), Recall (+14.55%) và F1-score (+14.14%). Điều này khẳng định rằng việc chuẩn hóa giúp giảm sai số dự đoán và làm tăng hiệu suất của mô hình.

Bên cạnh đó, bài tiểu luận cũng đã triển khai một giao diện ứng dụng đơn giản, cho phép người dùng nhập các thông tin về triệu chứng và nhận được kết quả dự đoán từ mô hình kNN đã được tối ưu hóa. Đây là một bước tiến quan trọng trong việc ứng dụng khai phá dữ liệu vào hỗ trợ y tế, giúp cảnh báo sớm nguy cơ mắc bệnh tiểu đường và hỗ trợ quyết định điều trị.

Mặc dù đạt được những kết quả khả quan, bài nghiên cứu vẫn còn một số hạn chế, chẳng hạn như chưa thử nghiệm với nhiều thuật toán khác để so sánh hiệu suất hoặc chưa tối ưu hóa thêm về mặt thời gian tính toán của kNN. Do đó, trong tương lai, có thể mở rộng nghiên cứu bằng cách thử nghiệm các mô hình khác như Random Forest, SVM hoặc Deep Learning, đồng thời cải thiện giao diện ứng dụng để phục vụ người dùng tốt hơn.

Tóm lại, bài tiểu luận đã chứng minh tiềm năng của khai phá dữ liệu trong lĩnh vực y tế, đặc biệt là trong việc hỗ trợ chẩn đoán sớm bệnh tiểu đường. Hy vọng rằng nghiên cứu này sẽ đóng góp một phần nhỏ vào sự phát triển của các ứng dụng trí tuệ nhân tạo trong chăm sóc sức khỏe cộng đồng.