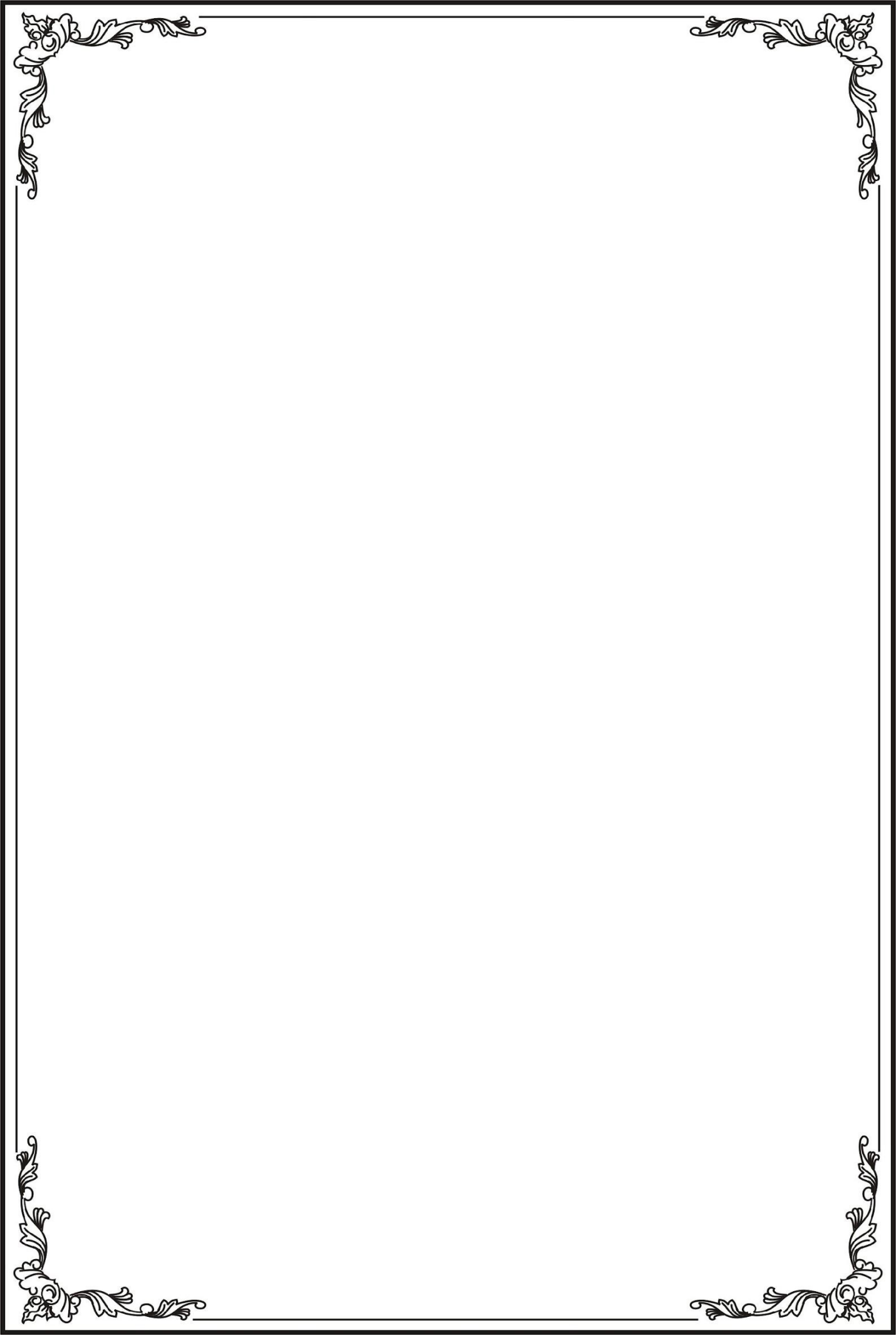
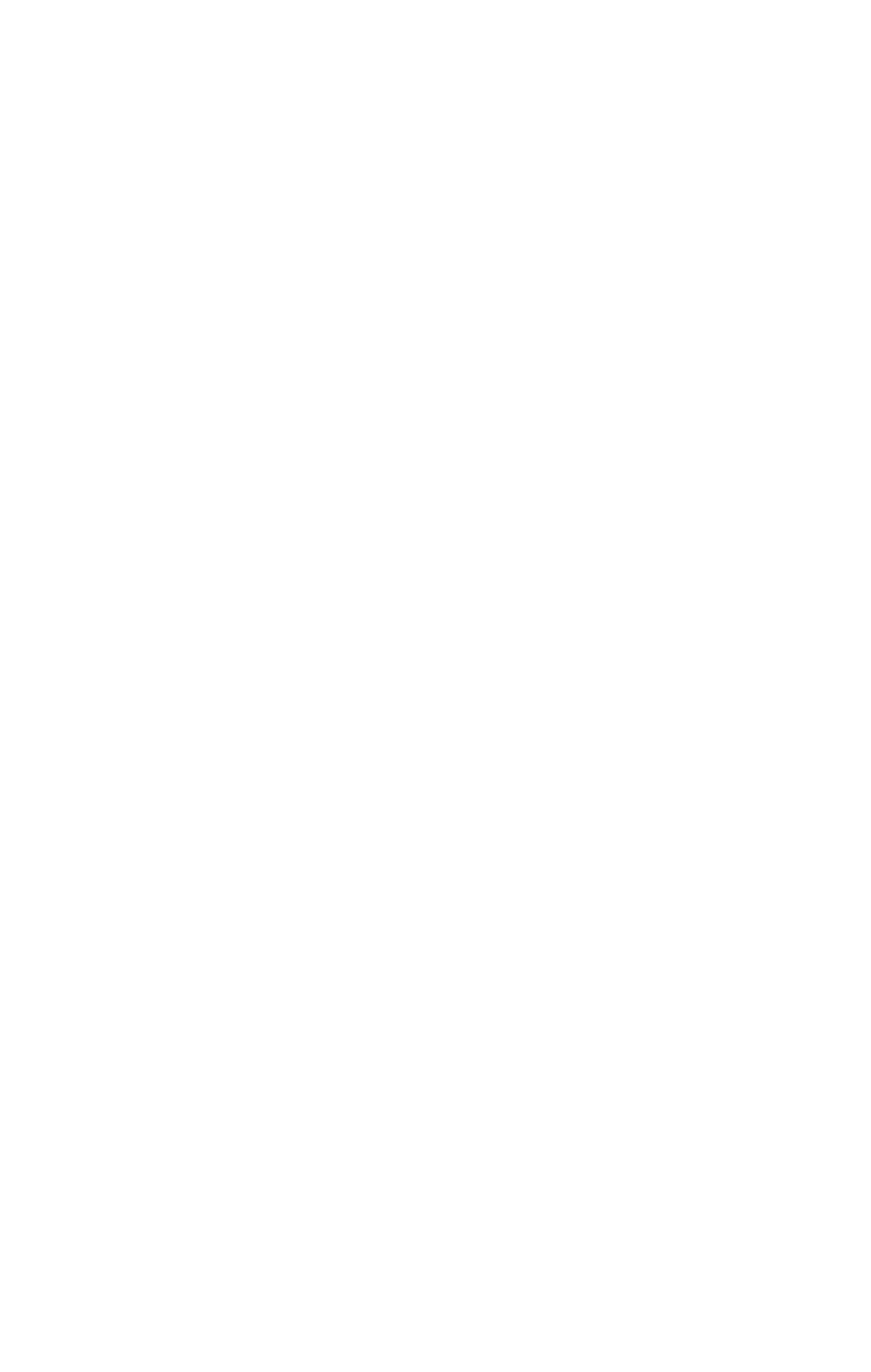
Z



UỶ BAN NHÂN DÂN TP. HỒ CHÍ MINH

**A blue circle with black text

Description automatically generatedTRƯỜNG ĐẠI HỌC SÀI GÒN**

BÁO CÁO CUỐI KỲ

MÔN HỌC:

NHẬP MÔN MÁY HỌC

ĐỀ TÀI:

**CHẨN ĐOÁN BỆNH TIỂU ĐƯỜNG**

**Giảng viên:**. ThS. Nguyễn Thanh Phước

**Thực hiện:**

3120410629 – Vũ Quốc Vương

3122410341 - Huỳnh Quang Quân

3121410566 - Nguyễn Chung Vũ Tùng

*TPHCM, ngày 07 tháng 12 năm 2024*

**NHẬN XÉT, ĐÁNH GIÁ CỦA GIẢNG VIÊN**

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

**MỤC LỤC**

**[Chương 1: Giới thiệu 1](#_Toc27538)**

[1.1. Giới thiệu đề tài 1](#_Toc17359)

[1.2. Mục đích 1](#_Toc31799)

[1.3. Định nghĩa bài toán 2](#_Toc18704)

[1.3.1. Dữ liệu đầu vào, dữ liệu đầu ra 2](#_Toc5122)

[1.3.2. Thuật toán sử dụng 2](#_Toc9127)

[1.3.3. Vai trò của khai phá dữ liệu trong việc giải quyết bài toán 2](#_Toc21625)

**[Chương 2: Mô tả dữ liệu 3](#_Toc20274)**

[2.1. Nguồn gốc bộ dữ liệu 3](#_Toc29225)

[2.2. Thống kê mô tả về dữ liệu 3](#_Toc24524)

**[Chương 3: Phương pháp nghiên cứu 4](#_Toc9088)**

[3.1. Các phương pháp phân tích dữ liệu 4](#_Toc16892)

[3.2. Cơ sở lý thuyết các mô hình được sử dụng 4](#_Toc15215)

[3.2.1. Logistic Regression 4](#_Toc23380)

[3.2.2. Random forest 5](#_Toc30235)

[3.2.3. Catboost 6](#_Toc6357)

**[Chương 4: Tiền xử lý dữ liệu 8](#_Toc28715)**

[4.1 Kiểm tra và xử lý dữ liệu trùng lặp 8](#_Toc8587)

[4.2 Kiểm tra và xử lý những dữ liệu bị thiếu 8](#_Toc17782)

[4.3 Xử lý những giá trị bất thường 9](#_Toc22277)

[4.3.1 Giá trị bị thiếu 9](#_Toc31254)

[4.3.2 Giá trị ngoại lai 10](#_Toc23189)

[4.4 Xử lý mất cân bằng giá trị: 11](#_Toc10547)

**[Chương 5: Phân tích khám phá dữ liệu 13](#_Toc17784)**

[5.1 Thông tin tổng quan 13](#_Toc3840)

[5.2 Thống kê tổng quan: 14](#_Toc29525)

[5.3 Phân tích đơn biến 16](#_Toc28604)

[5.3.1 Phân phối giá trị của từng đặc trưng 16](#_Toc31166)

[5.3.2 Boxplot của các đặc trưng 25](#_Toc11201)

[5.3.3 Phân tích các đặc trưng so với nhãn 28](#_Toc31886)

[5.4 Phân tích đa biến 35](#_Toc4312)

**[Chương 6: Thực nghiệm 37](#_Toc20055)**

[6.1. Chuẩn bị dữ liệu 37](#_Toc15219)

[6.2. Huấn luyện và đánh giá mô hình 37](#_Toc1352)

[6.2.1. Huấn luyện 37](#_Toc11901)

[6.2.2. Đánh giá mô hình 38](#_Toc12334)

[6.3. Cải thiện mô hình 48](#_Toc16094)

[6.4. So sánh kết quả chẩn đoán 53](#_Toc29814)

**[Chương 7: Cài đặt và sử dụng 57](#_Toc6118)**

[7.1. Cấu trúc dự án 57](#_Toc25955)

[7.2. Cài đặt 60](#_Toc2100)

[7.3. Sử dụng 60](#_Toc9685)

[7.4. Kiểm thử API 62](#_Toc740)

**[Chương 8: Kết quả 70](#_Toc12621)**

[8.1. Việc chênh lệch nhãn có ảnh hưởng đến mô hình chẩn đoán kết quả hay không? 70](#_Toc24319)

[8.2. Vì sao không loại bỏ hai đặc trưng có mức tương quan thấp với nhãn? 71](#_Toc19991)

**[Chương 9: Đánh giá và kết luận 73](#_Toc28537)**

[9.1. Đánh giá kết quả nghiên cứu 73](#_Toc31168)

[9.2. Những hạn chế trong nghiên cứu 73](#_Toc8279)

[9.3. Hướng nghiên cứu tiếp theo 73](#_Toc27201)

# **Chương 1: Giới thiệu**

* 1. **Giới thiệu đề tài**

Cùng với sự phát triển của thời đại, công nghệ thông tin ngày nay được sử dụng nhiều vào các lĩnh vực khác nhau, đóng vai trò quan trọng trong đời sống con người. Chính vì thế, lượng dữ liệu khổng lồ mới luôn được cập nhật liên tục, tạo điều kiện thuận lợi để nắm bắt xu hướng đổi mới. Như John Hagel III đã nói rằng "Dữ liệu là dầu mỏ của thời đại kỹ thuật số." câu nói này nhấn mạnh tầm quan trọng của dữ liệu, có dữ liệu chẳng khác gì có được nguồn tài nguyên quý giá.

Khai phá dữ liệu được ví như chìa khóa để mở ra cánh cửa tri thức, bởi vì tính thực tế của nó có thể giải quyết hầu hết các vấn đề của xã hội. Qua việc sử dụng thuật toán và những kỹ thuật tiên tiến để phân tích, thống kê những xu hướng của thời đại, chẩn đoán các sự kiện trong tương lai và còn nhiều mặt khác để thúc đẩy đổi mới, nâng cao chất lượng cuộc sống.

Trong lĩnh vực y tế, việc đưa ra những chẩn đoán nhanh chóng và chính xác là điều rất cần thiết, chính vì vậy con người đã tích hợp sử dụng trí tuệ nhân tạo (AI), học máy (machine learning) vào trong y tế và sử dụng rộng rãi. Nhờ hệ thống chẩn đoán của máy học trong y tế, thì việc nhận biết chẩn đoán bằng kỹ thuật thông thường đã được nâng cao đáng kể, giúp phát hiện được những căn bệnh tiềm tàng mà người bệnh có thể gặp phải, từ đó đưa ra đánh giá và cho ra kết quả điều trị kịp thời.

Theo thống kê của thế giới số người mắc bệnh tiểu đường năm 2024 dự kiến là 578 triệu người và có tỷ lệ tử vong là 7,1 triệu người, đây là một con số báo động vì tỷ lệ tử vong vẫn đang ở mức rất cao. Vì thế việc chẩn đoán bệnh tiểu đường bằng máy học trong y tế cần được đặt lên hàng đầu để đưa ra chẩn đoán chính xác để phát hiện bệnh tật sớm và đưa ra những phương hướng điều trị kịp thời. Nhờ đó đề tài chẩn đoán bệnh tiểu đường bằng máy học được nghiên cứu và đánh giá qua bài báo cáo sau đây.

* 1. **Mục đích**

Để cải thiện chất lượng sức khỏe của tất cả mọi người, thì việc đưa máy học vào y tế là điều thực sự cần thiết, bởi chính tiềm năng to lớn ấy sẽ tạo ra ra những tri thức dữ liệu lớn để phục vụ cho đời sống con người. Ngày nay, dữ liệu về y tế rất lớn và có xu hướng thay đổi rất nhiều, điều này dẫn đến dữ liệu ngày càng phức tạp và kèm theo nhiều thách thức. Đối diện với nhiều khó khăn đó chúng ta cần nghiên cứu nhiều mô hình máy học để đưa ra nhiều những chẩn đoán chính xác hơn.

Trong phạm vi của báo cáo, dữ liệu của người bệnh được đưa vào mô hình chẩn đoán bệnh tiểu đường, qua phương pháp phân tích thông kê mô tả thì những dự liệu về triệu chứng của người bệnh trước đó được huấn luyện để đưa ra chẩn đoán. Điều này hỗ trợ phát hiện, chẩn đoán sớm được nguy cơ mắc bệnh tiểu đường.

Tóm lại, báo cáo sẽ đề xuất các hướng phát triển và nghiên cứu tiếp theo trong lĩnh vực này. Điều này có thể bao gồm việc mở rộng phạm vi của mô hình để chẩn đoán các yếu tố nguy cơ khác liên quan đến bệnh tiểu đường, hoặc áp dụng mô hình vào các tình huống thực tế để kiểm tra tính ứng dụng và hiệu quả của nó trong thực tiễn y tế.

* 1. **Định nghĩa bài toán**
     1. **Dữ liệu đầu vào, dữ liệu đầu ra**
* Dữ liệu đầu vào
* Dữ liệu đầu vào của bài toán là số đo của các đặc trưng y tế.
* Dữ liệu đầu vào là dữ liệu số thực.
* Dữ liệu đầu ra
* Dữ liệu đầu ra là giá trị 0 hoặc 1 để xác định người đó có mắc bệnh tiểu đường (1) hay không mắc bệnh tiểu đường (0).
  + 1. **Thuật toán sử dụng**

Trong bài toán chẩn đoán khả năng mắc bệnh tiểu đường, các mô hình được sử dụng để đưa ra kết quả chẩn đoán là:

* Logistic Regression
* RF – Random Forest
* Catbootst
  + 1. **Vai trò của khai phá dữ liệu trong việc giải quyết bài toán**

Khai phá dữ liệu là giúp chúng ta hiểu rõ hơn về dữ liệu, từ đó tạo ra những phát hiện mới và đưa ra quyết định chính xác. Nói cách khác, khai phá dữ liệu giúp chúng ta khám phá ra những thông tin ẩn chứa trong dữ liệu và áp dụng nó vào nhiều lĩnh vực. Trong bài báo cáo này cũng áp dụng khai phá dữ liệu để huấn luyện mô hình đưa ra kết quả chẩn đoán bệnh tiểu đường.

**Chương 2: Mô tả dữ liệu**

**2.1. Nguồn gốc bộ dữ liệu**

Để đạt được hiệu quả trong việc phân tích và huấn luyện mô hình, tôi đã chọn nền tảng nổi tiếng Kaggle, nơi lưu trữ nhiều bộ dữ liệu khác nhau để tìm kiếm bộ dữ liệu phù hợp cho hướng nghiên cứu. Chính vì thế, bộ dữ liệu về bệnh tiểu đường sau đây cũng được tìm kiếm và lấy dữ liệu từ đó, với tên “Diabetes Dataset” đây là dữ liệu được chọn để đưa ra chẩn đoán.

**2.2. Thống kê mô tả về dữ liệu**

Đây là bộ dữ liệu có nguồn gốc từ Viện tiểu đường, tiêu hóa và bệnh thận quốc gia được nhiều người chọn lọc sử dụng, dữ liệu này được đăng tải lên website Kaggle với định dạng CSV.

- Bộ dữ liệu “Diabetes Dataset” chứa nhiều dữ liệu của bệnh nhân ẩn danh liên quan đến bệnh tiểu đường được các nhà nghiên cứu chia sẻ lên Kaggle để phân tích và đưa ra những khám phá mới về căn bệnh này.

- Bộ dữ liệu này bao gồm 768 mẫu bệnh nhân nữ ở Ấn Độ Pima có liên quan đến bệnh tiểu đường với 9 đặc trưng cần thiết để thể hiện nguyên nhân mắc bệnh, một đặc trưng trong đó là kết quả cho biết người đó có bị bệnh hay không.

|  |  |  |
| --- | --- | --- |
|  | **Đặc trưng** | **Thông tin** |
| 1 | Pregnancies | Số lần một người phụ nữ đã mang thai. |
| 2 | Glucose | Lượng đường trong máu hai giờ sau khi uống đồ uống có đường (xét nghiệm dung nạp glucose đường uống) |
| 3 | BloodPressure | Huyết áp tâm trương (mm Hg) |
| 4 | SkinThickness | Độ dày của nếp gấp da trên cơ tam đầu (cánh tay trên) (mm) |
| 5 | Insulin | Mức insulin trong máu sau hai giờ (mu U/ml). |
| 6 | BMI | Chỉ số khối cơ thể, thước đo cân nặng so với chiều cao (weight in kg/(height in m)^2) |
| 7 | DiabetesPedigreeFunction | Điểm được tính toán dựa trên tiền sử gia đình mắc bệnh tiểu đường. |
| 8 | Age | Tuổi của người phụ nữ tính theo năm. |
| 9 | Outcome | Bệnh nhân bị tiểu đường (1) hay không bị (0). |

***Bảng 1: Bảng thông tin các đặc trưng và nhãn***

**Chương 3: Phương pháp nghiên cứu**

**3.1. Các phương pháp phân tích dữ liệu**

Để hiểu rõ hơn vềbộ dữ liệu “Diabetes Dataset”, việc phân tích dữ liệu là điều đặc biệt cần thiết để rút ra những đánh giá về dữ liệu, từ đó đưa ra những quyết định để xử lý và tiến hành xây dựng mô hình. Trong báo cáo này, các phương pháp được để phân tích được cụ thể sau đây:

* **Thống kê mô tả dữ liệu:** được dùng để tóm tắt và giải thích dữ liệu một cách hiệu quả, bằng cách thống kê cơ bản các giá trị như trung bình (**mean**), trung vị (**median**), độ lệch chuẩn (**standard deviation**), các phân vị (**quartile, percentile**) và tần số (**frequency**). Những kết quả thống kê này làm rõ hơn về phân phối và các giá trị đặc trưng của từng biến.
* **Tính cân bằng của biến mục tiêu:** để đánh giá xem biến mục tiêu có cân bằng hay không, nếu dữ liệu nhãn bị lệch nhiều về một bên, sẽ tiến hành các kĩ thuật xử lý phù hợp.
* **Trực quan hóa dữ liệu:** sử dụng các biểu đồ như biểu đồ tần suất (**histogram**) để kiểm tra phân bố dữ liệu , biểu đồ hộp (**box plot**) dùng để phát hiện giá trị ngoại lai và biểu đồ tương quan (**heatmap**) để biết được độ tương quan tuyến tính giữa các đặc trưng với nhau. Những biểu đồ này giúp dễ dàng nhận biết mối tương quan giữa các biến để từ đó đưa ra những quyết định cho các bước xử lý sau để tăng độ hiệu quả cho mô hình.

**3.2. Cơ sở lý thuyết các mô hình được sử dụng**

**3.2.1. Logistic Regression**

**Logistic regression** hay còn được gọi là hồi quy logistic là một thuật toán học máy được sử dụng rộng rãi cho các vấn đề liên quan đến phân loại. Ở dạng cơ bản, nó được sử dụng cho bài toán phân loại nhị phân chỉ có hai lớp để chẩn đoán. Tuy nhiên, với một chút mở rộng và chất xám, logistic regression có thể dễ dàng được sử dụng cho vấn đề phân loại nhiều lớp.

Cở sở lý thuyết: Mô hình Logistic Regression dựa trên **hàm sigmoid** để ước lượng xác suất của mỗi lớp, với giá trị đầu ra nằm trong khoảng từ 0 đến 1. Hàm sigmoid có công thức như sau:

A graph of a function

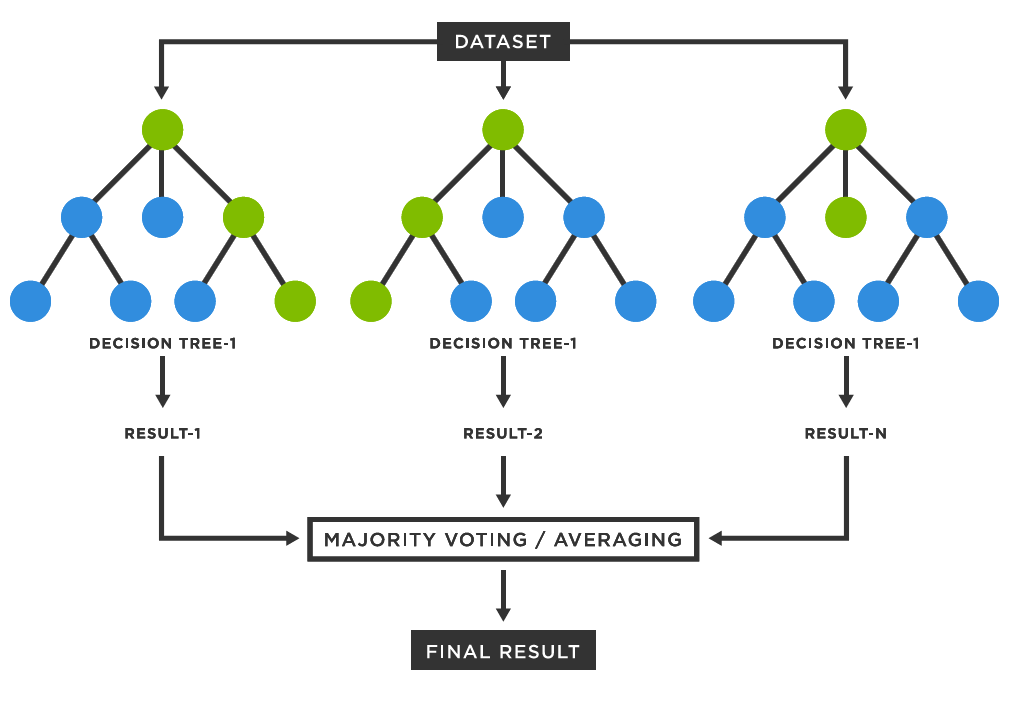
Description automatically generated*Hình 3.1: Hàm sigmoid trong Logistic regression*

Tiếp đó, để mục tiêu tối đa hóa khả năng chẩn đoán xác suất đúng, Logistic Regression tối ưu hóa hàm **log-likelihood** (dạng mất mát cho phân loại nhị phân):

Trong đó: S là tổ hợp tuyến tính của các đặc trưng đầu vào, hệ số chặn (bias term), là trọng số của đặc trưng và giá trị của đặc trưng. Còn đối với hàm log-likelihood thì là xác suất chẩn đoán.

**3.2.2. Random forest**

Random forest là một phương pháp thống kê mô hình hóa bằng máy (machine learning statistic) dùng để phục vụ các mục đích phân loại, tính hồi quy và các nhiệm vụ khác bằng cách xây dựng nhiều cây quyết định (Decision tree).

*Hình 3.2: Mô hình thuật toán Random Forest*

Cơ sở lý thuyết: Random Forest xây dựng nhiều cây quyết định trên các tập con ngẫu nhiên của dữ liệu huấn luyện (bootstrap sampling) và lựa chọn các đặc trưng ngẫu nhiên trong mỗi phân chia (feature bagging).

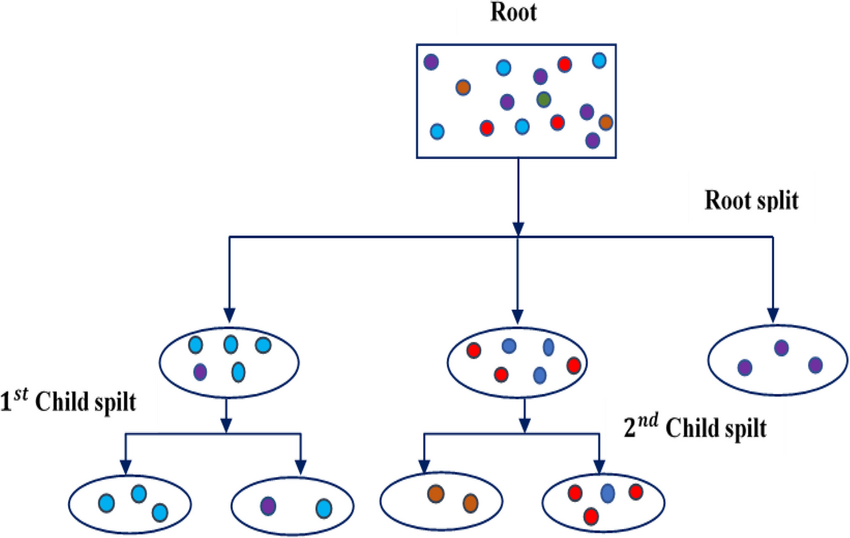
Kết quả phân loại được đưa ra bằng cách lấy nhãn có số bầu chọn nhiều nhất (**majority voting**):

và kết quả hồi quy: .

Trong đó được hiểu là nhãn chẩn đoán cây thứ n trong rừng, là giá trị chẩn đoán cây thứ i và T là số lượng cây trong rừng.

**3.2.3. Catboost**

CatBoost (Categorical Boosting) là một thuật toán machine learning dựa trên Cây quyết định tăng cường độ dốc (GBDT) được thiết kế để xử lý hiệu quả dữ liệu phân loại. Thuật toán này tối ưu hóa chẩn đoán bằng cách quản lý các tính năng phân loại và giá trị còn thiếu.



*Hình 3.3: Mô hình thuật toán CatBoost*

CatBoost hoạt động bằng cách tối ưu hóa các hàm mất **Log-loss** (phân loại nhị phân) có công thức: , với L là hàm mất mát (loss function), là giá trị thực tế và là giá trị chẩn đoán.

Cơ sở lý thuyết: CatBoost là một thuật toán boosting gradient, là một kỹ thuật học tập tổng hợp xây dựng một mô hình chẩn đoán bằng cách kết hợp các chẩn đoán của nhiều mô hình cơ sở, thường là cây quyết định. Nó hoạt động bằng cách tối ưu hóa hàm mất mát để giảm thiểu lỗi chẩn đoán. Mỗi cây học sẽ cố gắng sửa sai cho cây trước đó. Và CatBoost còn sử dụng một kỹ thuật đặc biệt gọi là "**ordered boosting**" để giảm bias trong quá trình huấn luyện, đồng thời cũng làm cho mô hình học một cách hiệu quả.

# **Chương 4: Tiền xử lý dữ liệu**

|  |
| --- |
| df = pd.read\_csv(**'diabetes.csv'**) |

## **4.1 Kiểm tra và xử lý dữ liệu trùng lặp**

Thông qua xem xét thì không tồn tại dữ liệu trùng lập.

|  |
| --- |
| duplicated\_rows = df[df.duplicated()] print(**"Các hàng trùng lặp:"**) print(duplicated\_rows) *# Xóa các hàng trùng* df = df.drop\_duplicates() |

A close-up of a computer screen

Description automatically generated

## **4.2 Kiểm tra và xử lý những dữ liệu bị thiếu**

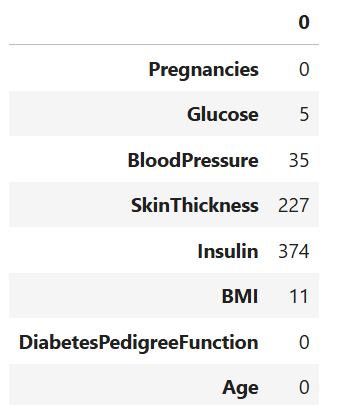
Dựa vào nguồn gốc của tập dữ liệu [1], càng chắc chắn hơn là từ Bệnh Viện Quốc gia về Bệnh tiểu đường, Tiêu hóa và Thận Bệnh tật (National Institute of Diabetes and Digestive and Kidney Diseases), lĩnh vực y tế, các số liệu 0 hay giá trị âm được xem là giá trị thiếu, do lỗi nhập thông tin hoặc không điền. Vậy nên, sẽ chuyển chúng thành NaN để dễ xử lý:

|  |
| --- |
| features = [**'Glucose'**, **'BloodPressure'**,**'SkinThickness'**,**'Insulin'**,**'BMI'**,**'Age'**] **for** feature **in** features:     df[feature] = df[feature].apply(**lambda** x: np.nan **if** x <= 0 **else** x) |

Từ tính toán:

|  |
| --- |
| numerical\_features = [**'Pregnancies'**, **'Glucose'**, **'BloodPressure'**, **'SkinThickness'**, **'Insulin'**, **'BMI'**, **'DiabetesPedigreeFunction'**, **'Age'**] df[numerical\_features].isnull().sum() |

Ta thấy rằng số lượng giá trị bị thiếu được thể hiện qua Figure 1.



*Figure 1:Bảng các thuộc tính với số lượng giá trị bị thiếu tương ứng*

Trước khi tiếp tục xử lý dữ liệu, xây dựng và huấn luyện mô hình thì nên chia dữ liệu thành tập huấn luyện (train) và tập kiểm tra (test). Để tránh rủi ro bị rò rỉ dữ liệu (data leakage).

|  |
| --- |
| train, test = train\_test\_split(df, test\_size=0.2, random\_state=42) |

## **4.3 Xử lý những giá trị bất thường**

### **4.3.1 Giá trị bị thiếu**

Cách để xử lý các dữ liệu này là sử dụng phương pháp điền giá trị (Imputation) để vá hết các giá trị thiếu của tập dữ liệu. Điều này là cần thiết để giúp cho mô hình học hỏi hiệu quả mà không bị phức tạp hóa hay nhiễu dữ liệu với những tập dữ liệu không có đủ thông tin rõ ràng.

Giá trị để điền là giá trị trung vị (Median), khi đối mặt với tập dữ liệu không được cung cấp đầy đủ thông tin quan trọng như Glucose, Insulin, có sự chênh lệch quá mức sự giữa số liệu thực tế, thì điền giá trị bằng trung vị (Median Imputation) được xem là một cách hợp lí với tập dữ liệu này. Trung vị là giá trị ở giữa của một tập dữ liệu được sắp xếp, nên nó không bị ảnh hưởng bởi các giá trị cực đoan, khiến nó trở thành lựa chọn tốt cho dữ liệu bị lệch.

Median sẽ được tính ở tập train, sau đó áp dụng giá trị đó đồng thời lên tập test, rồi cố định lại kiểu dữ liệu của từng đặc trưng.

|  |
| --- |
| train\_1 = train[train[**'Outcome'**]==1] train\_0 = train[train[**'Outcome'**]==0]  *# Use train set to calculate median* median\_1 = train\_1[numerical\_features].median() median\_0 = train\_0[numerical\_features].median()  *# Fill train set by applying median from train calculation* train.loc[train[**'Outcome'**]==1, numerical\_features] = train.loc[train[**'Outcome'**]==1, numerical\_features].fillna(median\_1) train.loc[train[**'Outcome'**]==0, numerical\_features] = train.loc[train[**'Outcome'**]==0, numerical\_features].fillna(median\_0)  *# Fill test set by applying median from train calculation* test.loc[test[**'Outcome'**]==1, numerical\_features] = test.loc[test[**'Outcome'**]==1, numerical\_features].fillna(median\_1) test.loc[test[**'Outcome'**]==0, numerical\_features] = test.loc[test[**'Outcome'**]==0, numerical\_features].fillna(median\_0)  **for** feature **in** numerical\_features:   train[feature] = train[feature].astype(dfCopy[feature].dtype)   test[feature] = test[feature].astype(dfCopy[feature].dtype) |

Trong tập dữ liệu này vì gặp phải quá nhiều số liệu bị lỗi đo lường chênh lệch so với số liệu thực tế nên chọn việc loại bỏ chúng để tránh gây bất ổn, làm giảm hiệu suất của mô hình. Giữ chúng lại có thể sẽ khiến cho dữ liệu bị phức tạp hóa, mô hình không thể học và tiếp thu được, chỉnh sửa chúng lại càng không nên vì điều đấy sẽ khiến dữ liệu có thể sẽ không thực tế và điều này nên được hạn chế.

Việc xử lý giá trị ngoại lai cần được lưu ý ở hiệu suất của mô hình để đảm bảo hiệu quả. Nếu việc xử lý chúng giúp tăng độ hiệu suất, thì nên xử lý, ngược lại thì không cần thiết.

### **4.3.2 Giá trị ngoại lai**

Giá trị ngoại lai là dữ liệu khác biệt đáng kể so với các quan sát khác trong một tập dữ liệu. Có thể xảy ra do sự thay đổi trong dữ liệu hoặc lỗi đo lường dẫn đến sự chênh lệch quá mức với số liệu thực tế, chúng có tác động lớn đến các phép đo thống kê, như đánh giá trung bình và ảnh hưởng đến hiệu suất mô hình, đặc biệt là trong các thuật toán nhạy cảm như RandomForest, Logistic Regression, Super Vector Machine, KNN.

Để khắc phục vấn đề này, có thể giữ, thay đổi hoặc loại bỏ giá trị ngoại lai tùy vào độ ảnh hưởng lên hiệu suất của mô hình. Sau đây là code loại bỏ giá trị ngoại lai theo từng hàng.

|  |
| --- |
| **def** remove\_outliers(train, test):     **for** feature **in** numerical\_features:         Q1 = train[feature].quantile(0.25)         Q3 = train[feature].quantile(0.75)         IQR = Q3 - Q1         lower\_bound = Q1 - 1.5 \* IQR         upper\_bound = Q3 + 1.5 \* IQR         train = train.loc[(train[feature] >= lower\_bound) & (train[feature] <= upper\_bound)]         test = test.loc[(test[feature] >= lower\_bound) & (test[feature] <= upper\_bound)]      **return** train, test  train, test = remove\_outliers(train, test) |

Trong tập dữ liệu này vì gặp phải quá nhiều số liệu bị lỗi đo lường chênh lệch so với số liệu thực tế nên chọn việc loại bỏ chúng để tránh gây bất ổn, làm giảm hiệu suất của mô hình. Giữ chúng lại có thể sẽ khiến cho dữ liệu bị phức tạp hóa, mô hình không thể học và tiếp thu được, chỉnh sửa chúng lại càng không nên vì điều đấy sẽ khiến dữ liệu có thể sẽ không thực tế và điều này nên được hạn chế.

Việc xử lý giá trị ngoại lai cần được lưu ý ở hiệu suất của mô hình để đảm bảo hiệu quả. Nếu việc xử lý chúng giúp tăng độ hiệu suất, thì nên xử lý, ngược lại thì không cần thiết.

## **4.4 Xử lý mất cân bằng giá trị:**

A screenshot of a computer

Description automatically generated

Như đã thấy ở trên, class 0 có số lượng gấp đôi class 1, cho thấy sự mất cân bằng giữa 2 class trong tập dữ liệu.

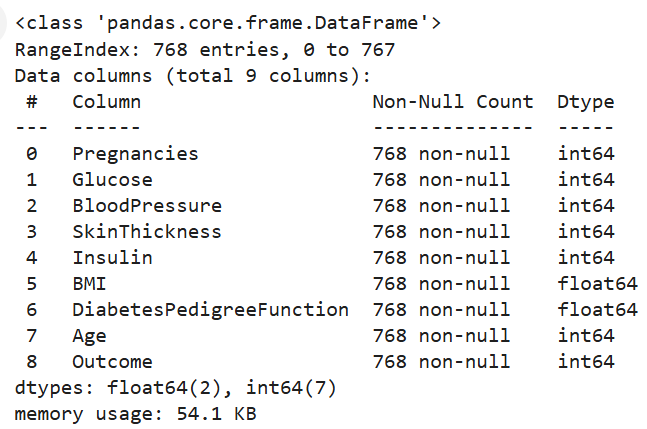
Để khắc phục vấn đề này, có thể sử dụng các phương pháp cân bằng dữ liệu như SMOTE (Synthetic Minority Over-sampling Technique) hay RUS (RandomUnderSampling, không khuyến khích), sau đây sẽ sử dụng SMOTE để cân bằng dữ liệu.

|  |
| --- |
| **from** imblearn.over\_sampling **import** SMOTE smote = SMOTE(random\_state=42) X\_train = train.drop(**'Outcome'**, axis=1) y\_train = train[**'Outcome'**] X\_train\_resampled, y\_train\_resampled = smote.fit\_resample(X\_train, y\_train) X\_train\_resampled[**'BMI'**] = X\_train\_resampled[**'BMI'**].round(1) X\_train\_resampled[**'DiabetesPedigreeFunction'**] = X\_train\_resampled[**'DiabetesPedigreeFunction'**].round(3) |

# **Chương 5: Phân tích khám phá dữ liệu**

## **5.1 Thông tin tổng quan**

Thông tin tổng quan của tập dữ liệu là thông tin về các thuộc tính của tập dữ liệu: tên thuộc tính và tập dữ liệu.



*Figure 3: Kết quả của info()*

Từ Figure 3 và phần mô tả ta của [1] có bảng sau:

|  |  |  |
| --- | --- | --- |
| Tên thuộc tính | Khoảng giá trị | Mô tả thuộc tính |
| Pregnancies | Số tự nhiên | Số lần mang thai |
| Glucose | Số tự nhiên | Nồng độ glucose huyết tương sau 2 giờ trong xét nghiệm dung nạp glucose đường uống |
| BloodPressure | Số tự nhiên | Huyết áp tâm trương (mm Hg) |
| SkinThickness | Số tự nhiên | Độ dày nếp gấp da cơ tam đầu (mm) |
| Insulin | Số tự nhiên | Insulin huyết thanh 2 giờ (mu U/ml) |
| BMI | Số thực không âm | Chỉ số khối cơ thể (cân nặng tính bằng kg/(chiều cao tính bằng m)^2) |
| DiabetesPedigreeFunction | Số thực không âm | Hàm phả hệ bệnh tiểu đường |
| Age | Số tự nhiên | Tuổi |
| Outcome | 0, 1 | Giá trị 1 là bị bệnh tiểu đường, giá trị 0 là không bị bệnh tiểu đường |

*Bảng 1: Thống kê mô tả thuộc tính*

## **5.2 Thống kê tổng quan:**

Thống kê tổng quan của tập dữ liệu là số lượng giá trị, giá trị trung bình (mean), độ lệch chuẩn (std), tứ phân vị thứ nhất (Q1 hay 25%), giá trị trung vị (Q2 hay 50%), tứ phân vị thứ ba (Q3 hay là 75%), giá trị cực tiểu (min), giá trị cực đại (max) của từng biến. Có thể thấy qua Figure 4 (có thể bỏ qua cột Outcome vì Outcome là biến phân loại).

A screenshot of a computer screen

Description automatically generated

*Figure 4 Kết quả từ describe()*

Nói thêm về Figure 4:

1. Pregnancies (Số lần mang thai)

Mean: Trung bình có khoảng 3.85 lần mang thai.

Std: Độ lệch chuẩn là 3.37, cho thấy sự biến động lớn trong số lượng thai kỳ giữa các bệnh nhân.

Min: Ít nhất là 0 lần (người chưa từng mang thai).

Max: Nhiều nhất là 17 lần.

2. Glucose (Nồng độ glucose)

Mean: Trung bình nồng độ glucose là 120.89 mg/dL.

Std: Độ lệch chuẩn là 31.97, cho thấy sự biến động lớn trong nồng độ glucose.

Min: Ít nhất là 0 mg/dL, điều này có thể là giá trị bị thiếu hoặc không hợp lệ.

Max: Nhiều nhất là 199 mg/dL.

3. BloodPressure (Huyết áp)

Mean: Trung bình huyết áp là 69.11 mm Hg.

Std: Độ lệch chuẩn là 19.36, cho thấy sự biến động lớn trong huyết áp.

Min: Ít nhất là 0 mm Hg, điều này có thể là giá trị bị thiếu hoặc không hợp lệ.

Max: Nhiều nhất là 122 mm Hg.

4. SkinThickness (Độ dày da)

Mean: Trung bình độ dày da là 20.54 mm.

Std: Độ lệch chuẩn là 15.95, cho thấy sự biến động lớn trong độ dày da.

Min: Ít nhất là 0 mm, điều này có thể là giá trị bị thiếu hoặc không hợp lệ.

Max: Nhiều nhất là 99 mm.

5. Insulin (Insulin)

Mean: Trung bình insulin là 79.80 µU/mL.

Std: Độ lệch chuẩn là 115.24, cho thấy sự biến động rất lớn trong mức insulin.

Min: Ít nhất là 0 µU/mL, điều này có thể là giá trị bị thiếu hoặc không hợp lệ.

Max: Nhiều nhất là 846 µU/mL.

6. BMI (Chỉ số khối cơ thể)

Mean: Trung bình BMI là 31.99.

Std: Độ lệch chuẩn là 7.88, cho thấy sự biến động trong BMI.

Min: Ít nhất là 0.078, điều này có thể là giá trị bị thiếu hoặc không hợp lệ.

Max: Nhiều nhất là 67.1.

7. DiabetesPedigreeFunction (Hàm phả hệ bệnh tiểu đường)

Mean: Giá trị trung bình của hàm là 0.47.

Std: Độ lệch chuẩn là 0.33, cho thấy sự biến động của giá trị của hàm.

Min: Ít nhất là 0.078.

Max: Nhiều nhất là 2.42.

8. Age (Tuổi)

Mean: Trung bình tuổi là 33.24.

Std: Độ lệch chuẩn là 11.76, cho thấy sự biến động trong tuổi.

Min: Ít nhất là 21 tuổi.

Max: Nhiều nhất là 81 tuổi.

9. Outcome (Kết quả xác định bệnh tiểu đường)

Mean: Trung bình là 0.35, cho thấy khoảng 35% bệnh nhân bị tiểu đường.

Std: Độ lệch chuẩn là 0.48, cho thấy sự biến động trong kết quả.

Min: Ít nhất là 0 (không bị tiểu đường).

Max: Nhiều nhất là 1 (bị tiểu đường).

## **5.3 Phân tích đơn biến**

### **5.3.1 Phân phối giá trị của từng đặc trưng**

Phần tram của từng nhãn được thể hiện qua Figure 2.

A pie chart with a red and blue circle

Description automatically generated

*Figure 2: Biểu đồ tỉ lệ phần trăm mỗi nhãn*

Phân phối giá trị của từng biến được thể hiện qua các Figure 5, Figure 6, Figure 7, Figure 8, Figure 9, Figure 10, Figure 11, Figure 12, Figure 13

A graph of a pregnancy

Description automatically generated with medium confidence

*Figure 5: Biểu đồ histogram của Pregnancies*

A graph of glucose

Description automatically generated

*Figure 6: Biểu đồ histogram của Glucose*

A graph of a graph showing a number of blue bars

Description automatically generated with medium confidence

*Figure 7: Biểu đồ histogram của BloodPressure*

A graph with blue bars

Description automatically generated

*Figure 8: Biểu đồ histogram của SkinThickness*

A graph of a patient

Description automatically generated

*Figure 9 : Biểu đồ histogram của Insulin*

A graph of a diagram

Description automatically generated

*Figure 10: Biểu đồ histogram của BMI*

A graph of a patient's level

Description automatically generated

*Figure 11: : Biểu đồ histogram của DiabetesPedigreeFunction*

A graph of age and age

Description automatically generated

*Figure 12: Biểu đồ histogram của Age*

A graph with blue lines

Description automatically generated

*Figure 13: Biểu đồ histogram của Outcome*

### **5.3.2 Boxplot của các đặc trưng**

Phân tích Figure 14:

**Pregnancies:** Dữ liệu đang có ngoại lai, bởi có những mẫu mang thai rất cao(trên 13 lần).

**Glucose:** Tồn tại giá trị ngoại lai bằng 0 (theo dữ liệu y học thì Glucose là không hợp lí).

**Bloodpressure:** Giá trị tối thiểu là 0 (theo dữ liệu y học thì Blood pressure là không hợp lí).

**Skinthickness:** Giá trị lên đến 100 mm là quá cao và Skin thickness trong y học cũng không cho phép bằng 0.

**Insulin:** có nhều mức insulin cao nên cần kiểm tra ngoại lai để chứng thực.

**BMI:** Khoảng từ 0 đến 70, giá trị trên 40 cần kiểm tra lại và các giá trị bằng 0 là không được phép.

**DiabetesPedigreeFunction:** Khoảng từ 0 đến 2.5, các giá trị trên 1 chỉ ra nguy cơ di truyền cao hơn.

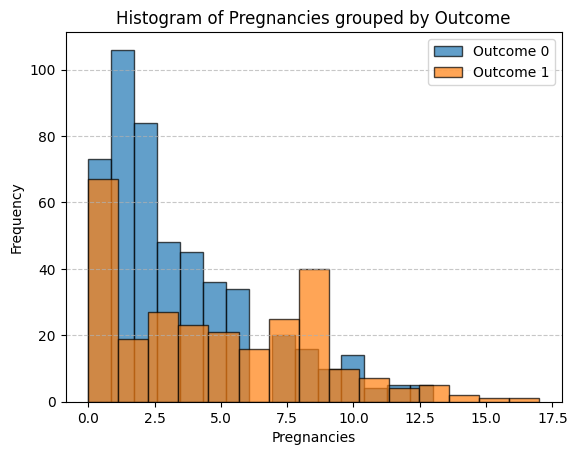
A screenshot of a graph

Description automatically generated

*Figure 14: Boxplot của từng thuộc tính*

### **5.3.3 Phân tích các đặc trưng so với nhãn**

Phân phôi giá trị của từng biến theo giá trị của Outcome được thể hiện qua : Figure 15, Figure 16, Figure 18, Figure 19, Figure 20, Figure 21, Figure 22



*Figure 15: Histogram của Pregnancies nhóm theo Outcome*

A graph of glucose

Description automatically generated

*Figure 16: Histogram của Glucose nhóm theo Outcome*

A graph of blood pressure

Description automatically generated

*Figure 17: Histogram của BloodPressure nhóm theo Outcome*

A graph of skin thickness

Description automatically generated

*Figure 18: Histogram của SkinThickness nhóm theo Outcome*

A graph of a number of patients with diabetes

Description automatically generated

*Figure 19: Histogram của Insulin nhóm theo Outcome*

A graph of a bar graph

Description automatically generated with medium confidence

*Figure 20: Histogram của BMI nhóm theo Outcome*

A graph of diabetes and diabetes

Description automatically generated

*Figure 21: Histogram của DiabetesPedigreeFunction nhóm theo Outcome*

A graph of a number of age groups

Description automatically generated

*Figure 22: Histogram của Age nhóm theo Outcome*

## **5.4 Phân tích đa biến**

Ma trận tương quan được thể hiện qua Figure 23

Từ Figure 23, ta có vài phân tích

Phát hiện một số **đặc trưng kém quan trọng** với Outcome là **BloodPressure**, **SkinThickness** và **Insulin**. Đồng thời cũng tìm ra được các đặc trưng **quan trọng** tuyến tính là **Glucose**, **BMI**, **Pregnancies**, **DiabetesPedigreeFunction**.

Tương quan giữa **Pregnancies và Age** là (0.54) là khá cao cần chú ý để **tránh đa cộng tuyến** trong mô hình.

A colorful squares with black and white text

Description automatically generated

*Figure 23: Ma trận tương quan*

**Chương 6: Thực nghiệm**

**6.1. Chuẩn bị dữ liệu**

Để cho chuyện nhầm lẫn hay sai sót không xảy ra khi đang viết lệnh, thì sau khi dữ liệu được trải qua tiền xử lý (được thực hiện ở ***Chương 4)***, đổi lại tên biến dữ liệu huấn luyện về nguyên trạng và gán đặc trưng phân loại “Outcome” cho biến kiểm tra y, đặc trưng còn lại cho biến kiểm tra x.

|  |
| --- |
| X\_train = X\_train\_resampled  y\_train = y\_train\_resampled  X\_test = test.drop(**'Outcome'**, axis=1) y\_test = test[**'Outcome'**] |

**6.2. Huấn luyện và đánh giá mô hình**

**6.2.1. Huấn luyện**

**a) RandomForest**

|  |
| --- |
| **from** sklearn.ensemble **import** RandomForestClassifier  rf = RandomForestClassifier(random\_state=42) rf.fit(X\_train, y\_train) |

Để sử dụng thuật toán/mô hình RandomForest thì phải import mô đun thuật toán RandomForestClassifier từ thư viện tổ hợp của sklearn.

**from** sklearn.ensemble **import** RandomForestClassifier

Sử dụng tham số random\_state=42 (hoặc số nào đó tùy thuộc vào người xây dựng) cho mô hình, thiết lập một hạt giống cho trình tạo số ngẫu nhiên được sử dụng trong thuật toán để đảm bảo khả năng tái tạo cố định kết quả qua mỗi lần chạy.

Đưa dữ liệu huấn luyện vào trong thuật toán.

rf = RandomForestClassifier(random\_state=42)  
rf.fit(X\_train, y\_train)

**b) Logistic**

|  |
| --- |
| **from** sklearn.linear\_model **import** LogisticRegression  lr = LogisticRegression(max\_iter=100000, random\_state=42) lr.fit(X\_train, y\_train) |

Tương tự với thuật toán/mô hình RandomForest, để sử dụng thuật toán/mô hình LogisticRegression cần phải import mô đun từ thư viện mô hình tuyến tính của sklearn.

**from** sklearn.linear\_model **import** LogisticRegression

Khác với RandomForest, không chỉ có tham số random\_state mà còn có max\_iter=100000, tham số này số lần lặp tối đa cho thuật toán tối ưu hóa hữu ích khi xử lý các tập dữ liệu yêu cầu nhiều lần lặp hơn để hội tụ để được hiệu quả tốt, bắt buộc có với các dữ liệu phức tạp, nếu để tham số bé hơn như 100 có thể sẽ dẫn đến không hội tụ và xuất hiện lỗi số lần lặp không thỏa mãn.

lr = LogisticRegression(max\_iter=100000, random\_state=42)  
lr.fit(X\_train, y\_train)

**c) CatBoost**

* **Yêu cầu tải thư viện catboost**

|  |
| --- |
| pip install catboost |

|  |
| --- |
| **from** catboost **import** CatBoostClassifier  cb = CatBoostClassifier(silent=**True**, random\_state=42) cb.fit(X\_train, y\_train) |

Thuật toán/Mô hình này khác xa với 2 thuật toán/mô hình trước đó, để sử dụng nó cần phải import mô đun CatBoostClassifier từ thư viện khác không phải là sklearn, đó chính là catboost.

**from** catboost **import** CatBoostClassifier

Ngoài tham số random\_state đã biết trước đó, silent là một tham số khiến bất kỳ thông báo hoặc cảnh báo nào trong quá trình đào tạo hoặc chẩn đoán sẽ không được in ra, hiển thị.

cb = CatBoostClassifier(silent=**True**, random\_state=42)  
cb.fit(X\_train, y\_train)

**6.2.2. Đánh giá mô hình**

**a) Threshold (Độ Ngưỡng)**

Threshold là giá trị mức độ ngưỡng mà tại đó mô hình phân loại đạt được hiệu suất tốt nhất theo một tiêu chí đánh giá cụ thể, ví dụ như **F1-score**, **Precision**, **Recall**. Đây là một phương pháp cực kỳ lợi ích và đóng vai trò cực kỳ lớn cho việc đánh giá mô hình, thay vì sử dụng ngưỡng mặc định là 0.5, có thể điều chỉnh ngưỡng này để phù hợp với mục tiêu của bài toán.

**Xây dựng trực quan hóa Threshold:**

|  |
| --- |
| y\_proba = cb.predict\_proba(X\_test)[:, 1]  thresholds = np.arange(0.1, 1.0, 0.1)  f1\_scores = []  **for** threshold **in** thresholds:     y\_pred = (y\_proba >= threshold).astype(int)     f1 = f1\_score(y\_test, y\_pred)     f1\_scores.append(f1)      print(**f"Threshold: {**threshold**:.2f}, F1-score: {**f1**:.4f}"**)  plt.plot(thresholds, f1\_scores, label=**"F1-score"**) plt.xlabel(**"Threshold"**) plt.ylabel(**"F1-score"**) plt.title(**"F1-score for Different Thresholds"**) plt.legend() plt.show() |

Tính xác suất chẩn đoán: y\_proba = cb.predict\_proba(X\_test)[:, 1]

* cb là tên mô hình CatBoost (có thể thay với lr, rf tùy vào mô hình đang sử dụng).
* predict\_proba trả về xác suất chẩn đoán cho từng lớp 0 và 1.
* [:, 1] để lấy xác suất thuộc lớp 1, vì phân loại chỉ có 2 lớp nên sẽ ưu tiên lớp dương tính để đảm bảo cân bằng.

Tạo danh sách các ngưỡng (thresholds) với np.arange(0.1, 1.0, 0.1) để xét từng ngưỡng lần lượt 0.1 đơn vị.

Lập vòng lặp qua mỗi threshold và so sánh giữa xác suất lớp 1 với threshold kiểu dữ liệu số nguyên. Nếu xác suất lớp 1 cao hơn threshold => chẩn đoán là lớp 1, ngược lại => chẩn đoán là lớp 0, điểm f1 sẽ được cập nhật qua từng vòng lặp:

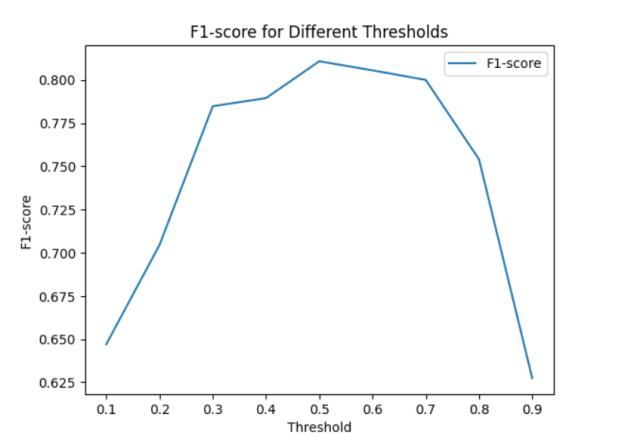
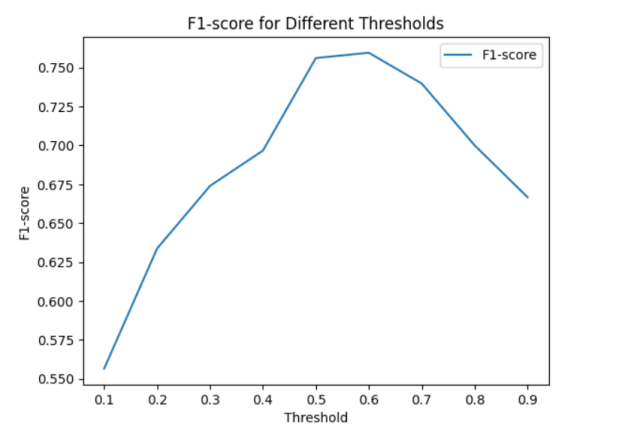
f1\_scores = []  
  
**for** threshold **in** thresholds:  
    y\_pred = (y\_proba >= threshold).astype(int)  
    f1 = f1\_score(y\_test, y\_pred)  
    f1\_scores.append(f1)  
  
    print(**f"Threshold: {**threshold**:.2f}, F1-score: {**f1**:.4f}"**)

Sau đó, tạo biểu đồ plot để trực quan hóa biểu đồ đường cho threshold:

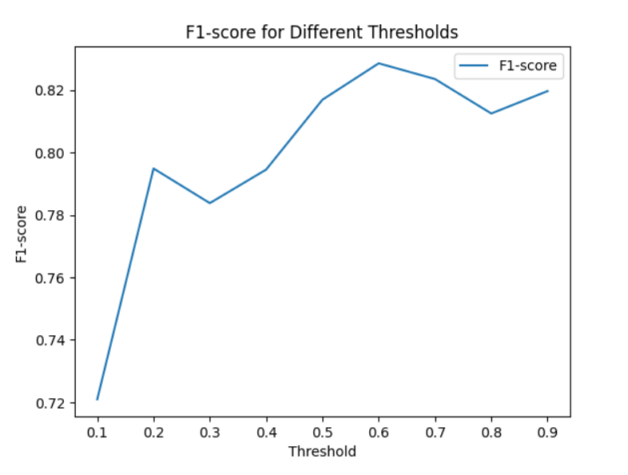
plt.plot(thresholds, f1\_scores, label=**"F1-score"**)  
plt.xlabel(**"Threshold"**)  
plt.ylabel(**"F1-score"**)  
plt.title(**"F1-score for Different Thresholds"**)  
plt.legend()  
plt.show()

* Các tham số trong plot lần lượt là danh sách threshold, danh sách điểm f1 và nhãn của đường thẳng trong biểu là “F1-score”.
* Label của x, y lần lượt là “Threshold” và “F1-score”.
* plt.legend() sẽ hiển thị một bảng nhỏ như là một chú thích hiển thị ý nghĩa của đường thẳng trong biểu đồ.

**Kết quả trực quan hóa Threshold:**



*Hình 6.1. LogisticRegression Hình 6.2. RandomForestClassifier*



*Hình 6.3. CatBoostClassifier*

Nhận xét kết quả trực quan hóa Threshold, chọn threshold phù hợp nhất của từng mô hình:

* RandomForestClassifier: 0.5
* LogisticRegression: 0.6
* CatBoostClassifier: 0.6

**b) Chi tiết chỉ tiêu đánh giá mô hình**

**b.1) Với tập kiểm tra**

Định nghĩa:

* **True Positive (TP):** số lượng đối tượng của lớp *positive* được phân loại đúng là *positive*.
* **True Negative (TN):** số lượng đối tượng của lớp *negative* được phân loại đúng là *negative*.
* **False Positive (FP):** số lượng đối tượng của lớp *negative* bị phân loại nhầm thành *positive*.
* **False Negative (FN):** số lượng đối tượng của lớp *positive* bị phân loại nhầm thành *negative*

**Accuracy (Độ chính xác của dữ liệu kiểm tra):**

Accuracy là tỷ lệ phần trăm chẩn đoán đúng cho tập kiểm tra. Nó có thể được tính toán dễ dàng bằng cách chia số lần chẩn đoán đúng cho tổng số lần chẩn đoán.

**Precision (Độ chính xác phân loại lớp dương tính):**

Precision là tỷ lệ chẩn đoán đúng trong số tất cả **các mẫu mà mô hình chẩn đoán** là dương tính.

Tập trung vào độ chính xác của chẩn đoán lớp dương tính. Phù hợp với bài toán nếu việc chẩn đoán nhầm lớp dương có hậu quả nghiêm trọng.

Precision càng cao, lớp 0 bị chẩn đoán nhầm là lớp 1 (false positives) sẽ càng thấp.

Nhưng nếu Recall thấp, nghĩa là lớp 1 được chẩn đoán đúng là lớp 1 (true positives) thấp, thì ngược lại với nó, lớp 1 bị chẩn đoán nhầm là lớp 0 (false negatives) cao. Vì mô hình sẽ chẩn đoán mẫu thực tế là lớp 0 rất nhiều.

Tóm lại: Threshold thấp và Recall thấp => Precision cao vì mô hình sẽ chẩn đoán mẫu là lớp 0 nhiều => false negatives cao.

**Recall (Thu hồi):**

Recall là tỷ lệ chẩn đoán đúng trong số tất cả **các mẫu thực sự** thuộc lớp dương tính.  
Tập trung vào khả năng tìm được tất cả các mẫu dương tính thực sự. Phù hợp với các bài toán nếu việc bỏ sót các mẫu dương tính là không thể chấp nhận.

Recall càng cao, lớp 1 bị chẩn đoán nhầm là lớp 0 (false negatives) sẽ càng thấp, lớp 1 được chẩn đoán đúng là lớp 1 (true positives) cao.

Nhưng nếu precision thấp, nghĩa là lớp 0 bị chẩn đoán nhầm là lớp 1 (false positives) cao. Mô hình đã chẩn đoán mẫu thực tế là lớp 1 rất nhiều.

Tóm lại: Precision thấp và Threshold thấp => Recall cao vì mô hình chẩn đoán mẫu là lớp 1 nhiều => false positives cao.

**F1-score:**

F1-score, là *harmonic mean (giá trị trung bình hài hòa)* của precision và recall, được dùng để đánh giá cân bằng hiệu suất nếu 2 chỉ tiêu Precision và Recall có sự chênh lệch không rõ ràng hoặc khó lựa chọn.

Đây là chỉ tiêu thường được chú ý sử dụng nhất nếu không xác định được mục tiêu bài toán, hoặc muốn xem xét hiệu suất của mô hình nào tốt hơn khi sự lựa chọn quá khó để quyết định.

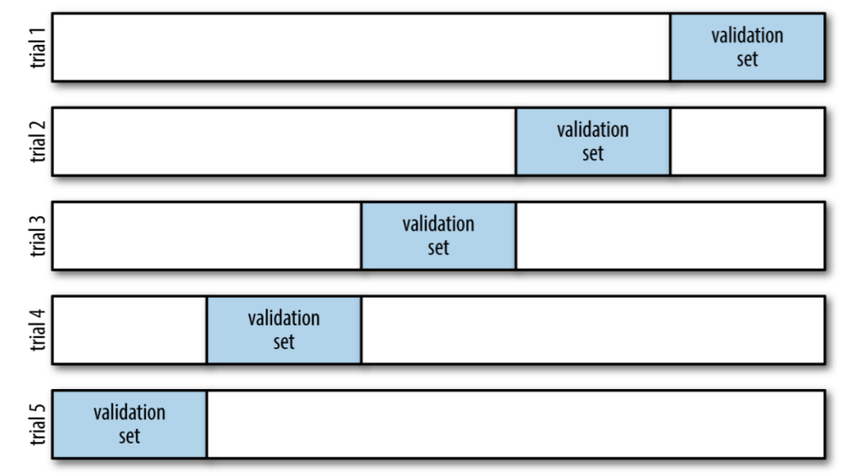
F1-score phù hợp trong bài toán mất cân bằng nhãn trong khi accuracy có thể đánh giá sai khi lớp chiếm đa số áp đảo.

Precision hoặc Recall có thể ưu tiên một mặt (giảm FP hoặc FN), nhưng F1-score cân bằng cả hai.

**b.2) Với tập huấn luyện**

**Cross-validation Score:**

Cross-validation score là kết quả của quá trình kiểm tra chéo (**cross-validation**) trên tập dữ liệu huấn luyện, đánh giá bằng cách sử dụng xác thực chéo k-fold. Nó biểu thị hiệu suất trung bình của mô hình qua nhiều lần lặp, trong đó tập dữ liệu được chia ngẫu nhiên thành k tập con (fold) và mô hình được đào tạo và thử nghiệm trên mỗi fold.



*Hình 6.4. Cách hoạt động của Cross-validation*

**b.3) Overfitting và Underfitting.**

Để xác định xem mô hình có bị Overfitting hay Underfitting không thì sẽ sử dụng Cross-validation Accuracy từ tập huấn luyện và Accuracy của tập kiểm tra.

Nếu, wps > 0.1:

* CV Accuracy > Accuracy của tập kiểm tra => Overfitting
* CV Accuracy < Accuracy của tập kiểm tra => Underfitting hoặc rò rĩ dữ liệu (data leakage)

**c) Thực hiện đánh giá mô hình**

**Chuẩn bị chỉ tiêu đánh giá:**

Chuẩn bị các chỉ tiêu đánh giá:

|  |
| --- |
| y\_pred = (y\_proba >= 0.5).astype(int) scores\_accuracy = cross\_val\_score(rf, X\_train, y\_train, cv=5, n\_jobs=-1, scoring= **"accuracy"**) accuracy = accuracy\_score(y\_test, y\_pred) precision = precision\_score(y\_test, y\_pred) recall = recall\_score(y\_test, y\_pred) f1 = f1\_score(y\_test,y\_pred) |

Sử dụng y\_proba đã được tạo ở ***Huấn luyện mô hình***, chuyển đổi chẩn đoán y\_proba thành chẩn đoán nhị phân y\_pred với ngưỡng xác suất rơi vào lớp 1 phải lớn hơn 0.6 (bé hơn sẽ rơi vào lớp 0).

cross\_val\_score thực hiện k-fold cross-validation trên tập dữ liệu, tham số lần lượt mô hình (rf, lr hoặc cb, tên mô hình ở ***Huấn luyện mô hình***), tập huấn luyện x\_train và y\_train, cv=5 chia tập huấn luyện ra thành 5 fold xác thực chéo, n\_jobs=-1 để vận dụng cpu hết mức tối đa, lựa chọn chỉ tiêu đánh giá accuracy:

cross\_val\_score(rf, X\_train, y\_train, cv=5, n\_jobs=-1, scoring= **"accuracy"**)

Tính accuracy, precision, recall, f1 lần lượt lên tập kiểm tra, tham số lần lượt là tập kiểm tra và tập chẩn đoán.

accuracy = accuracy\_score(y\_test, y\_pred)  
precision = precision\_score(y\_test, y\_pred)  
recall = recall\_score(y\_test, y\_pred)  
f1 = f1\_score(y\_test,y\_pred)

**Trực quan hóa:**

|  |
| --- |
| plt.figure(figsize=(7, 5)) plt.plot(range(1, 6), scores\_accuracy, marker=**'o'**, color=**'green'**, label=**"CV Accuracy"**) plt.axhline(np.mean(scores\_accuracy), color=**'green'**, linestyle=**'--'**, label=**"Mean CV Accuracy"**) plt.title(**"Cross-Validation Accuracy Scores"**) plt.xlabel(**"Fold"**) plt.ylabel(**"Accuracy"**) plt.legend() plt.tight\_layout()  plt.show() |

*Cross\_validation của tập huấn luyện*

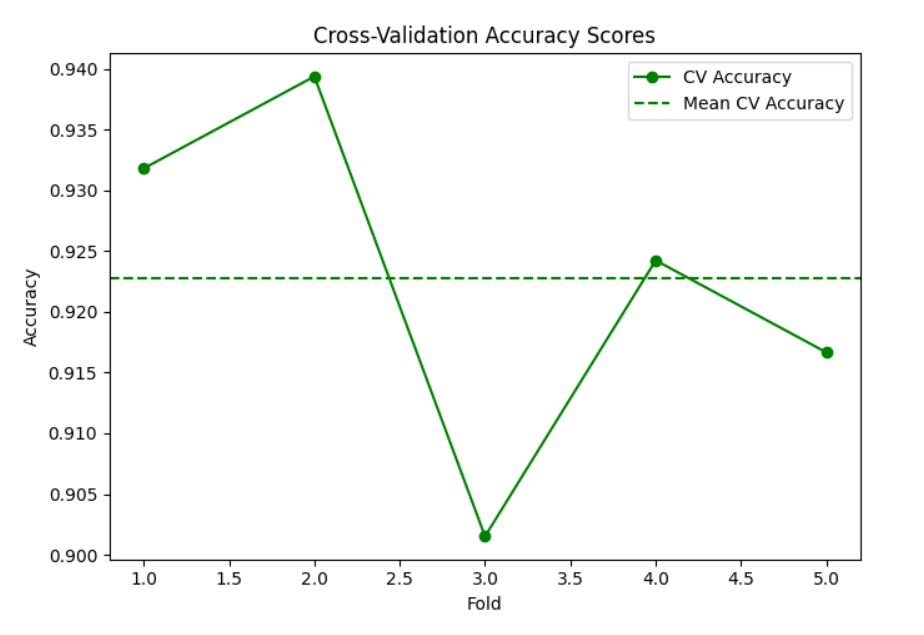
* Vẽ biểu đồ thể hiện chính xác Accuracy trên từng fold trong cross-validation, các tham số lần lượt là số thứ tự của các fold (1 đến 5), mảng chứa Accuracy của từng fold scores\_accuracy, giá trị trung bình Accuracy trên các fold np.mean(scores\_accuracy).
* plt.plot: Vẽ đường biểu diễn Accuracy trên từng fold.
* plt.axhline: Vẽ đường ngang thể hiện giá trị trung bình Accuracy.
* plt.legend(): Bảng chú thích đường thẳng.

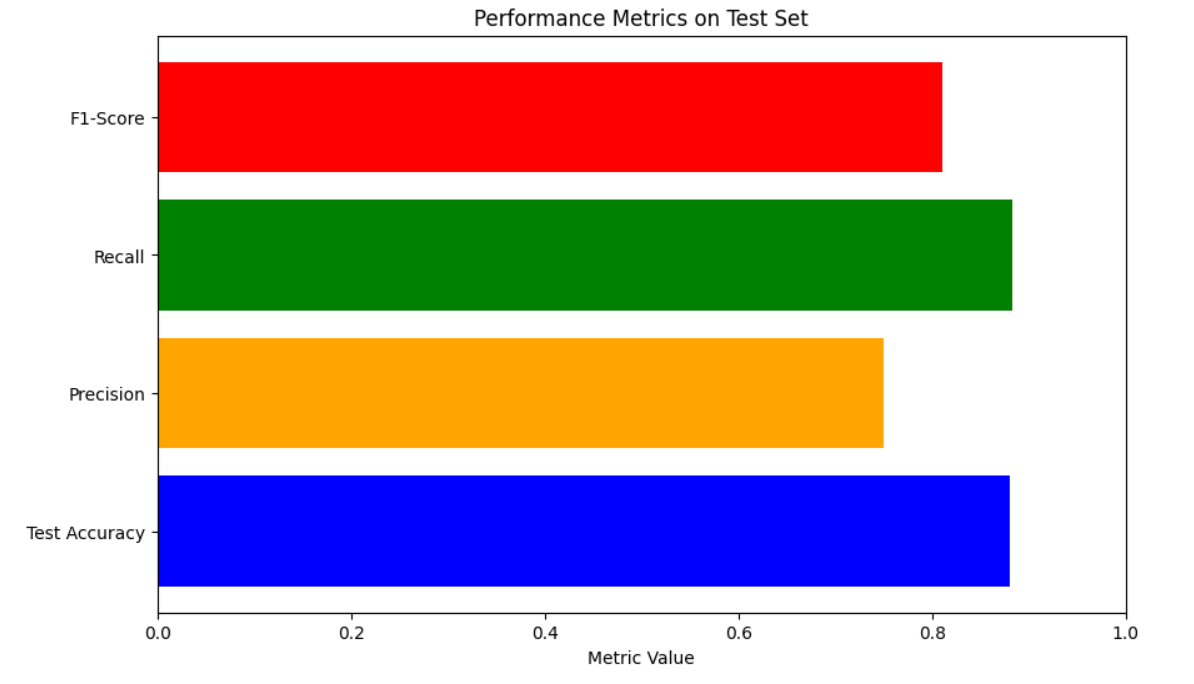
|  |
| --- |
| metrics = {     **"Test Accuracy"**: accuracy,     **"Precision"**: precision,     **"Recall"**: recall,     **"F1-Score"**: f1, }  fig, ax2 = plt.subplots(figsize=(10, 6)) metric\_names = list(metrics.keys()) metric\_values = list(metrics.values())  ax2.barh(metric\_names, metric\_values, color=[**'blue'**, **'orange'**, **'green'**, **'red'**]) ax2.set\_title(**"Performance Metrics on Test Set"**) ax2.set\_xlabel(**"Metric Value"**) ax2.set\_xlim(0, 1) |

*Các chỉ tiêu đánh giá tập kiểm tra*

* Tạo một từ điển metrics để trữ các chỉ tiêu đánh giá.
* Vẽ biểu đồ biểu đồ thanh ngang (horizontal bar chart) để hiển thị các chỉ số hiệu suất trong metrics.
* metric\_names: Danh sách tên các chỉ số (key của metrics).
* metric\_values: Danh sách giá trị các chỉ số (value của metrics).
* ax2.barh: Vẽ biểu đồ thanh ngang với tên chỉ số trên trục y và giá trị trên trục x.
* color=['blue', 'orange', 'green', 'red', 'purple']: Gán màu sắc khác nhau cho các thanh lần lượt dựa trên thứ tự metrics.
* ax2.set\_title và ax2.set\_xlabel: Thiết lập tiêu đề và nhãn cho trục x.
* ax2.set\_xlim(0, 1): Giới hạn trục x từ 0 đến 1 (vì các chỉ số nằm trong khoảng này).

**RandomForestClassifier:**





*Hình 6.5. Chỉ tiêu đánh giá mô hình RandomForest*

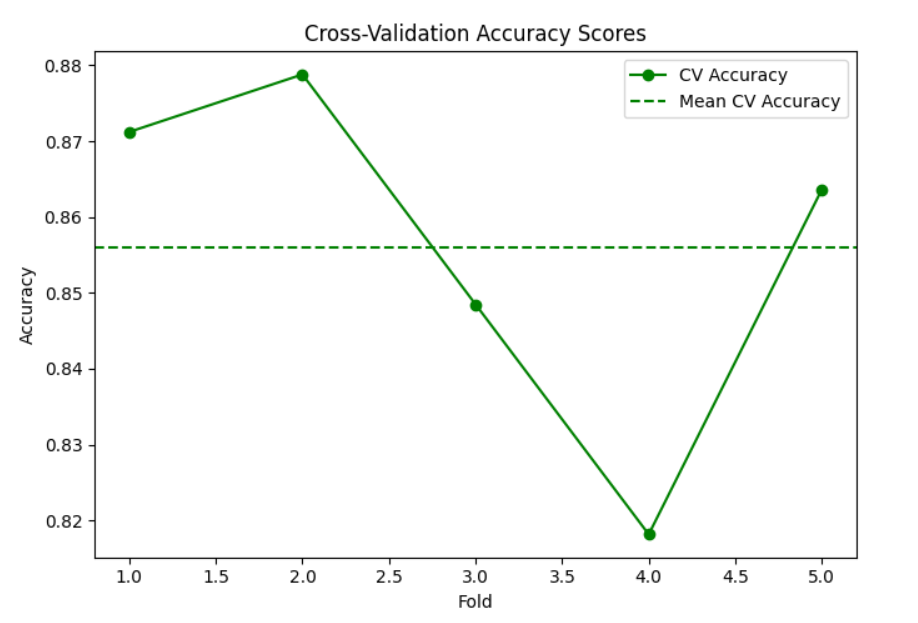
Kết quả đánh giá của mô hình:

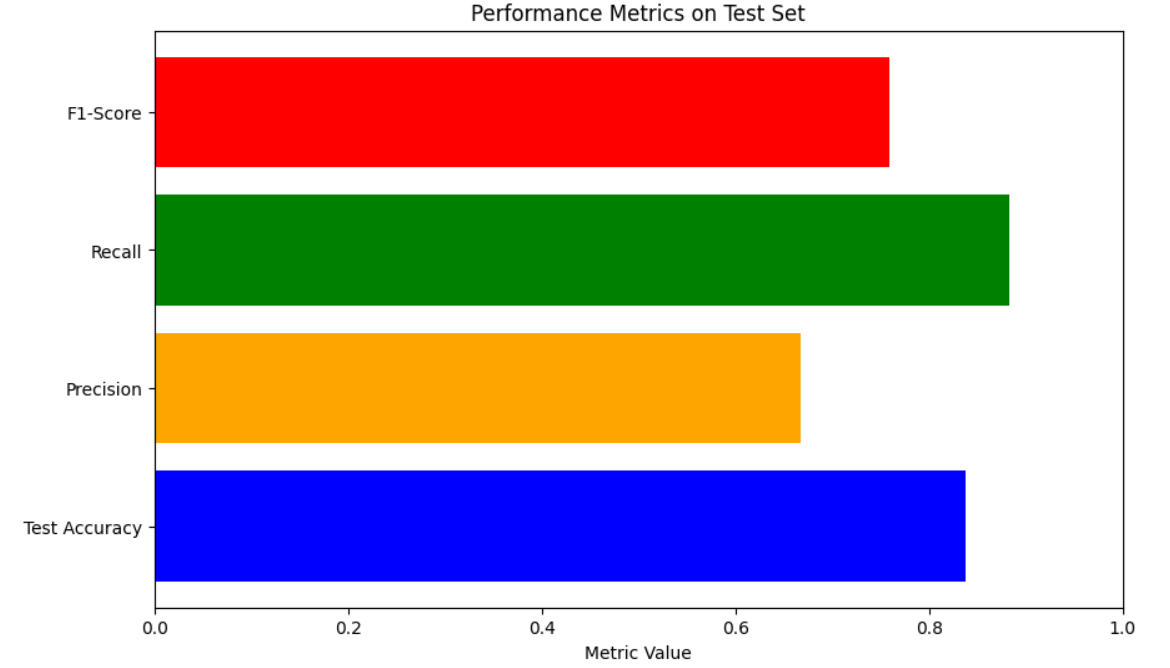
* Average(CV) Accuracy: 0.9227.
* Accuracy on Test: 0.8803.
* Precision: 0.7500.
* Recall: 0.8824.
* F1-score: 0.8108.

Đánh giá:

* CV > Test và |0.9227-0.8803| < 0.1 => Mô hình không bị Overfitting.
* Khoảng cách chênh lệch CV qua từng fold rất nhỏ, không ảnh hưởng lớn => Mô hình tiếp thu rất tốt.
* Recall ổn chứng tỏ mô hình phân loại được nhiều lớp 1, Precision khá ổn trong việc chẩn đoán chính xác nhiều lớp 1, F1 tóm lại được độ hiệu quả, hiệu suất tốt của mô hình.

**LogisticRegression:**





*Hình 6.6. Chỉ tiêu đánh giá mô hình Logistic*

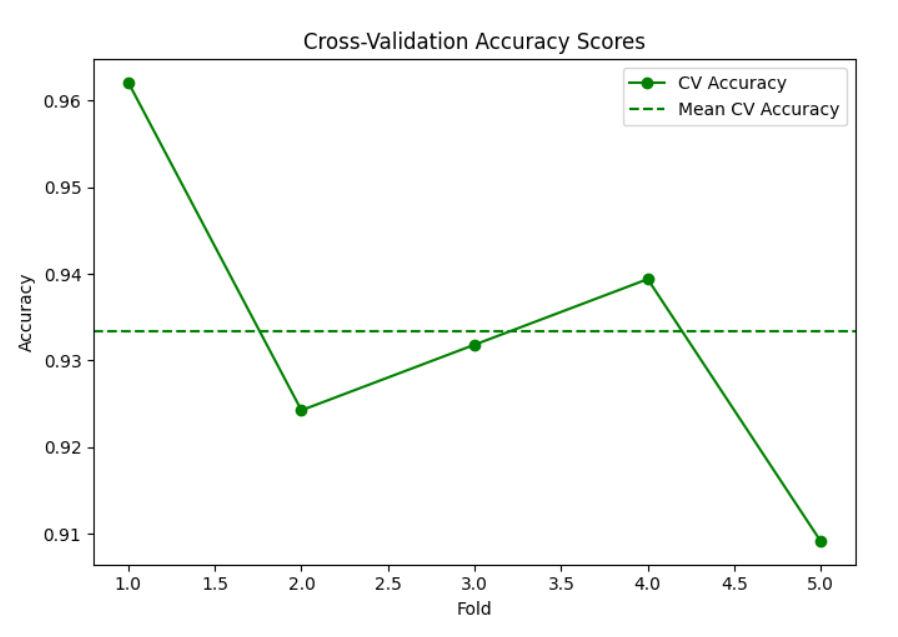
Kết quả đánh giá của mô hình:

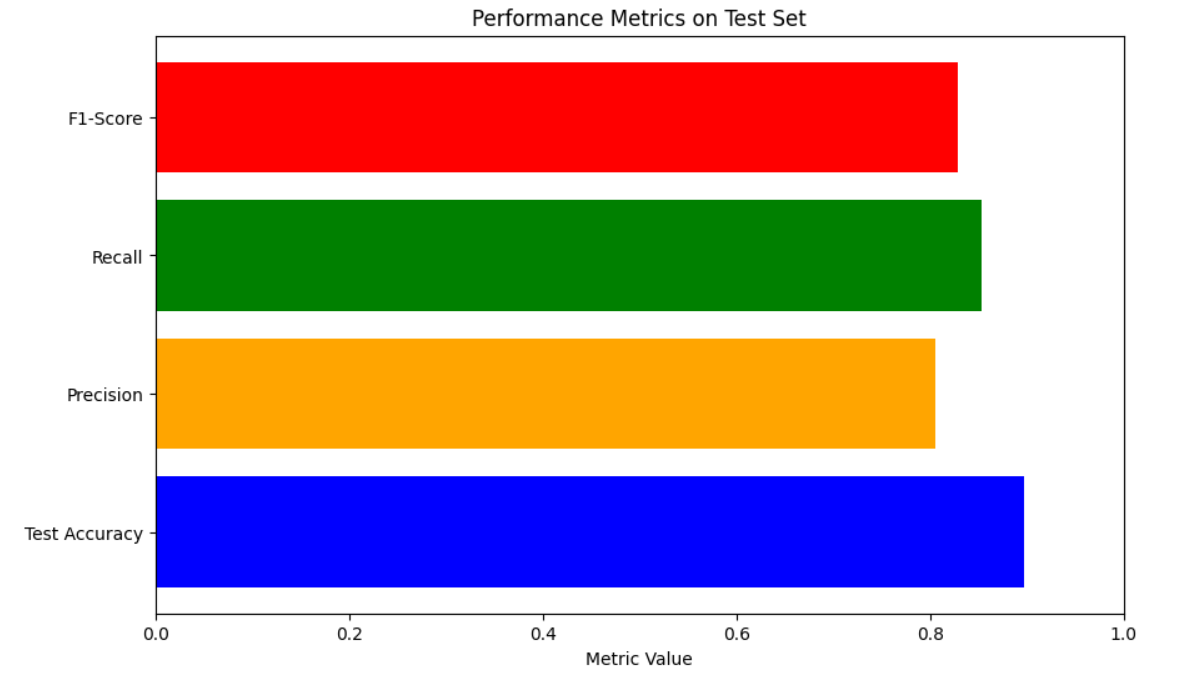
* Average(CV) Accuracy: 0.8561
* Accuracy on Test: 0.8376
* Precision: 0.6667
* Recall: 0.8824
* F1-score: 0.7595

Đánh giá:

* CV > Test và |0.8561-0.8376| < 0.1 => Mô hình không bị Overfitting.
* Khoảng cách chênh lệch CV qua từng fold nhỏ, không ảnh hưởng lớn => Mô hình tiếp thu tốt.
* Recall ổn chứng tỏ mô hình phân loại được nhiều lớp 1, Precision không được cao trong việc chẩn đoán chính xác nhiều lớp 1, F1 tóm lại được độ hiệu quả, hiệu suất khá ổn của mô hình, nhưng vẫn thua so với RandomForest.

**CatBoostClassifier:**





*Hình 6.7. Chỉ tiêu đánh giá mô hình CatBoost*

Kết quả đánh giá của mô hình:

* Average(CV) Accuracy: 0.9333
* Accuracy on Test: 0.8974
* Precision: 0.8056
* Recall: 0.8529
* F1-score: 0.8286

Đánh giá:

* CV > Test và |0.9333-0.8974| < 0.1 => Mô hình không bị Overfitting.
* Khoảng cách chênh lệch CV qua từng fold nhỏ, không ảnh hưởng lớn => Mô hình tiếp thu rất tốt. Nhưng sự chính xác qua từng fold giảm xuống khá nhiều dẫn đến sự quan ngại về hiệu suất của mô hình (không ổn định).
* Recall, Precision đều ổn trong việc phân loại và chẩn đoán chính xác nhiều lớp 1, F1 tóm lại được độ hiệu quả, hiệu suất tốt của mô hình.

**6.3. Cải thiện mô hình**

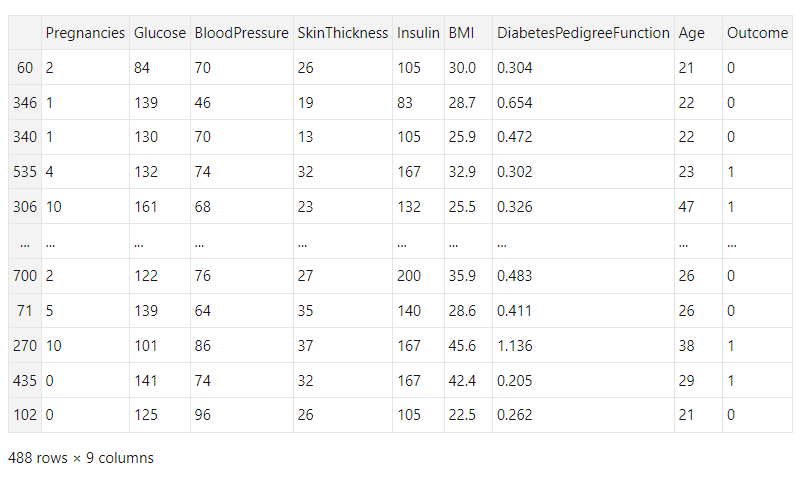
**a) Đổi cách xử lý loại bỏ ngoại lai**

Ở phần *Giá trị ngoại lai* của Chương *Tiền xử lý dữ liệu*, hàm cũ xử lý loại bỏ ngoại lai theo hàng (column\_wise), loại bỏ tất cả các mẫu theo hàng có giá trị ngoại lai có lẽ đã khiến cho mô hình không được phát huy tốt cũng như không tiếp thu được các đặc điểm quan trọng. Bởi vậy nên sẽ đổi cách xử lý loại bỏ ngoại lai, không xét theo hàng nữa mà xét theo thuộc tính (attribute\_wise).

Cách xử lý loại bỏ ngoại lai này sẽ giúp giữ lại các hàng khác không có giá trị ngoại lai trong thuộc tính đó, giảm thiểu mất dữ liệu. Cụ thể thì đấy là loại bỏ giá trị ngoại lai theo từng thuộc tính. Nhưng bù lại nếu một hàng có nhiều ngoại lai trên các thuộc tính khác nhau, các giá trị ngoại lai này vẫn có thể tồn tại trong dữ liệu sau khi lọc:

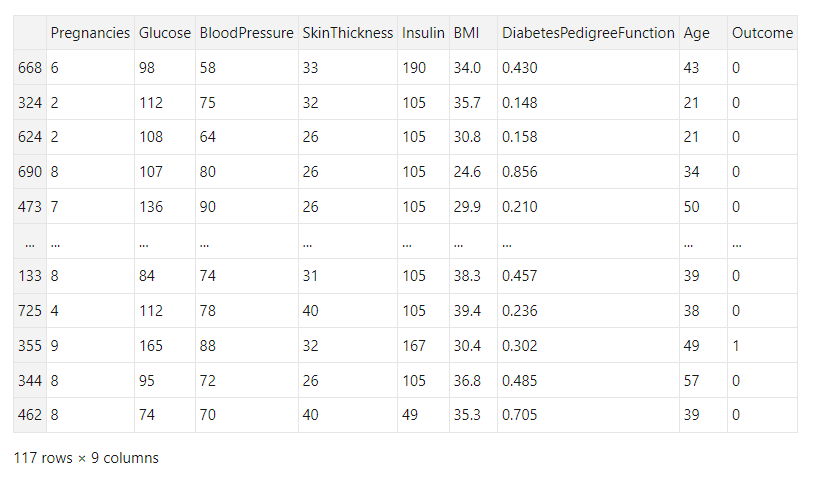
|  |
| --- |
| **def** remove\_outliers(train,test):     attributes = train.columns.to\_list()     outlierIndexes\_train = set()     outlierIndexes\_test = set()     **for** attribute **in** attributes:         Q1 = train[attribute].quantile(0.25)         Q3 = train[attribute].quantile(0.75)         IQR = Q3 - Q1         Min = Q1 - 1.5 \* IQR         Max = Q3 + 1.5 \* IQR         outlierIndexesOfAtrribute\_train = train[(train[attribute] < Min) | (train[attribute] > Max)].index.to\_list()         outlierIndexesOfAtrribute\_test = test[(test[attribute] < Min) | (test[attribute] > Max)].index.to\_list()         outlierIndexes\_train = outlierIndexes\_train.union(set(outlierIndexesOfAtrribute\_train))         outlierIndexes\_test = outlierIndexes\_test.union(set(outlierIndexesOfAtrribute\_test))     print(**"-------Loại bỏ giá trị ngoại lai hoàn tất!"**)     **return** train.drop(list(outlierIndexes\_train)), test.drop(list(outlierIndexes\_test)) train,test = remove\_outliers(train,test) |

Tập huấn luyện:



*Hình 6.8. Bảng dữ liệu tập huấn luyện*

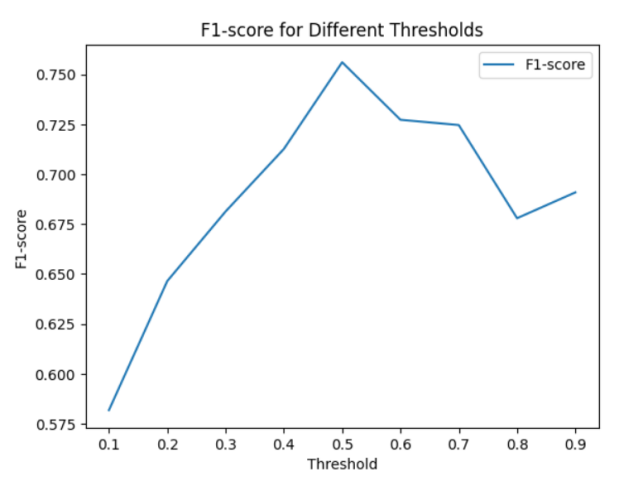
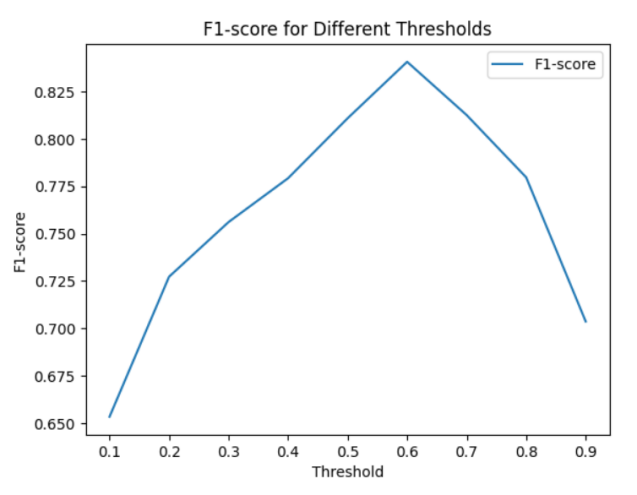
Tập kiểm tra:



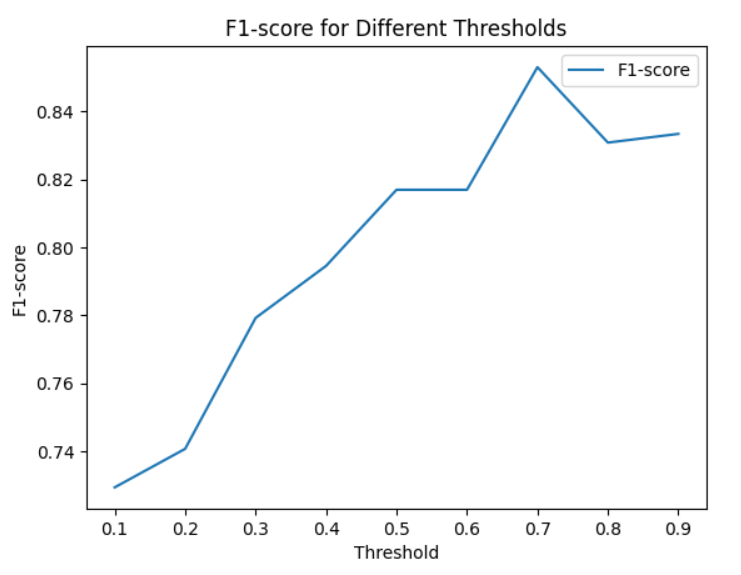
*Hình 6.9. Bảng dữ liệu tập huấn luyện*

Dữ liệu sau khi loại bỏ ngoại lai theo thuộc tính so với dữ liệu sau khi loại bỏ ngoại lai theo hàng cho thấy một vài ngoại lai đã được giữ lại, dễ thấy nhất là tập huấn luyện đã giữ lại 6 mẫu có giá trị ngoại lai, tập kiểm tra không có sự thay đổi gì.

**b) Đánh giá mô hình sau cải thiện**



*Hình 6.10. RandomForestClassifier Hình 6.11. LogisticRegression*

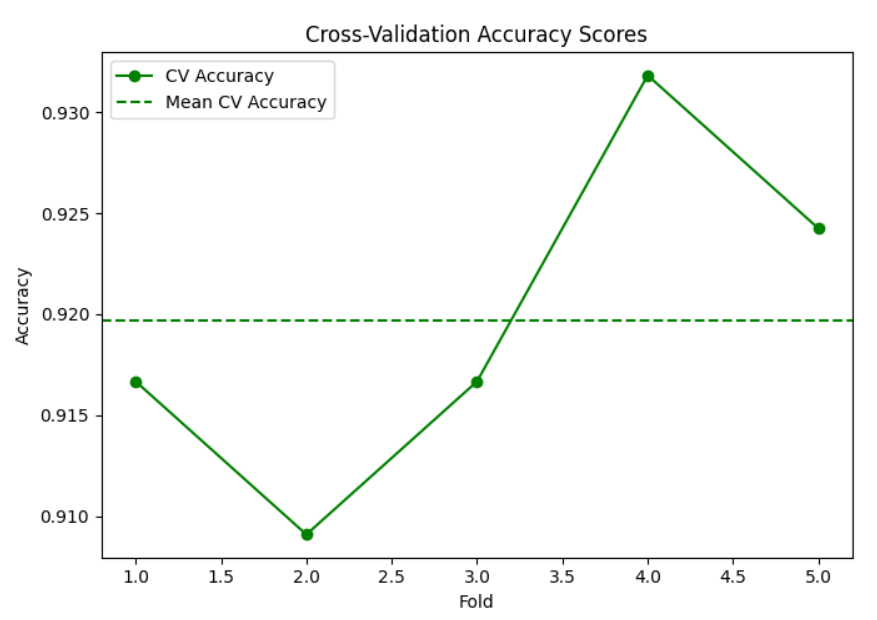


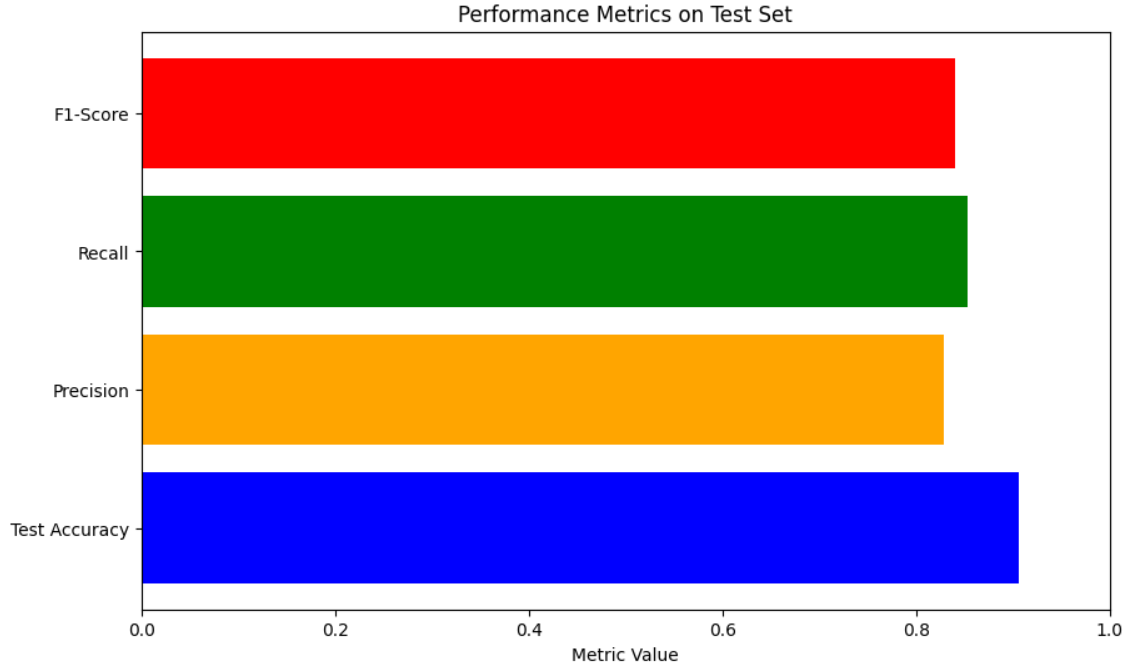
*Hình 6.12. CatBoostClassifier*

Ngưỡng (Threshold) của từng mô hình được cập nhật:

* RandomForestClassifier: 0.6
* LogisticRegression: 0.5
* CatBoostClassifier: 0.7

**RandomForestClassifier:**





*Hình 6.13. Chỉ tiêu của mô hình cải thiện RandomForest*

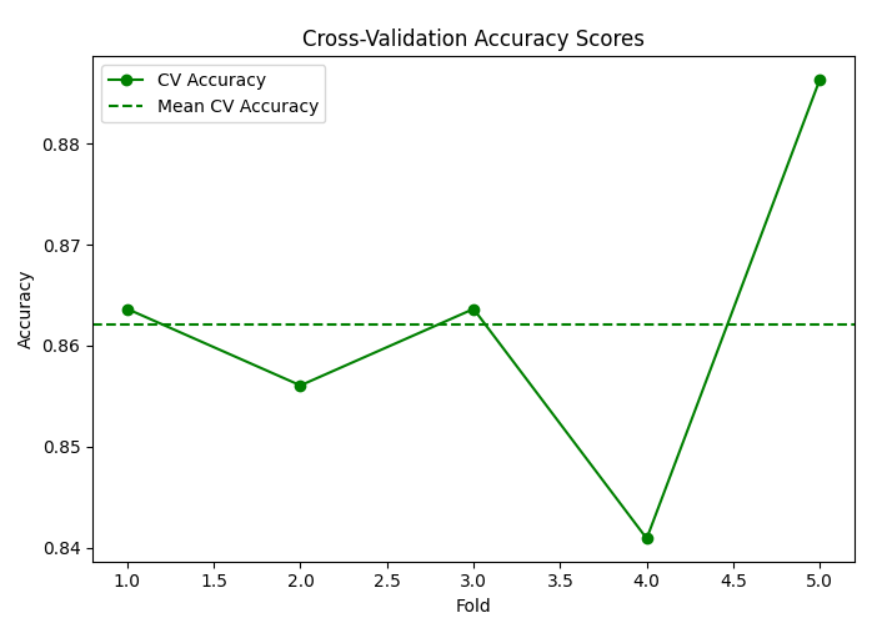
Kết quả đánh giá của mô hình:

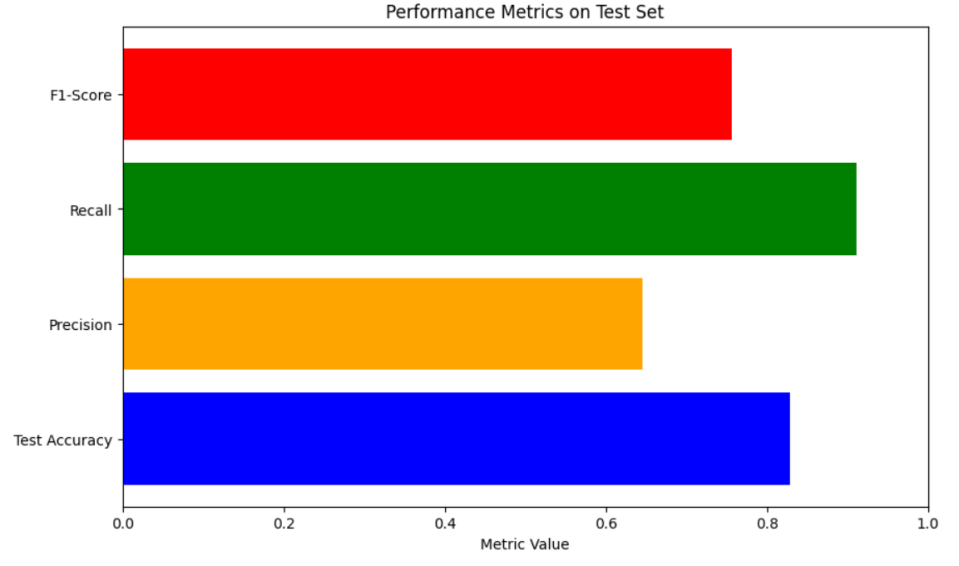
* Average(CV) Accuracy: 0.9197
* Accuracy on Test: 0.9060
* Precision: 0.8286
* Recall: 0.8529
* F1-score: 0.8406

Đánh giá:

* CV > Test và |0.9197-0.9060| < 0.1 => Mô hình không bị Overfitting.
* Khoảng cách chênh lệch CV qua từng fold rất nhỏ, không ảnh hưởng lớn => Mô hình tiếp thu rất tốt.
* Recall, Precision rất cân đối rất tốt trong việc phân loại và chẩn đoán chính xác lớp 1, F1 tóm lại được độ hiệu quả, hiệu suất rất ổn của mô hình.

**LogisticRegression:**





*Hình 6.14. Chỉ tiêu của mô hình cải thiện Logistic*

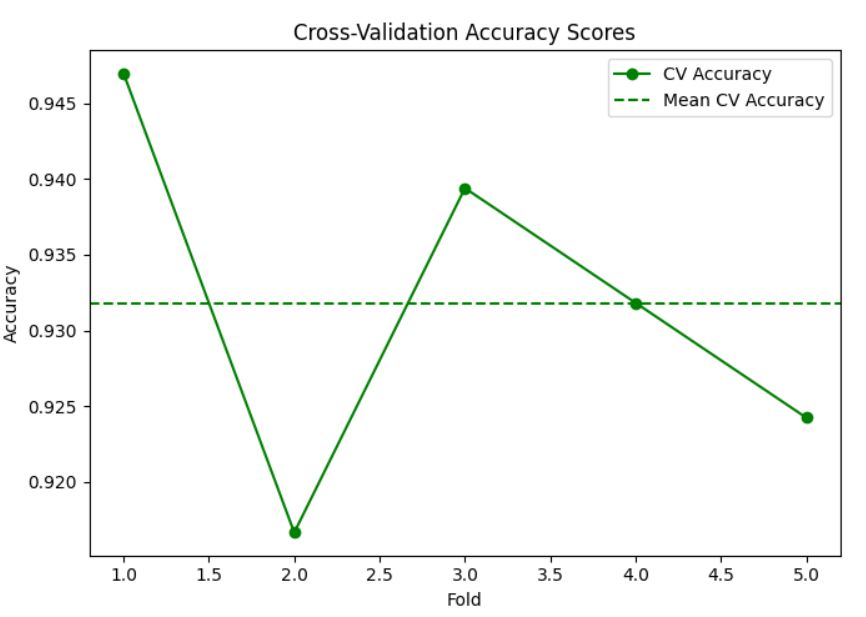
Kết quả đánh giá của mô hình:

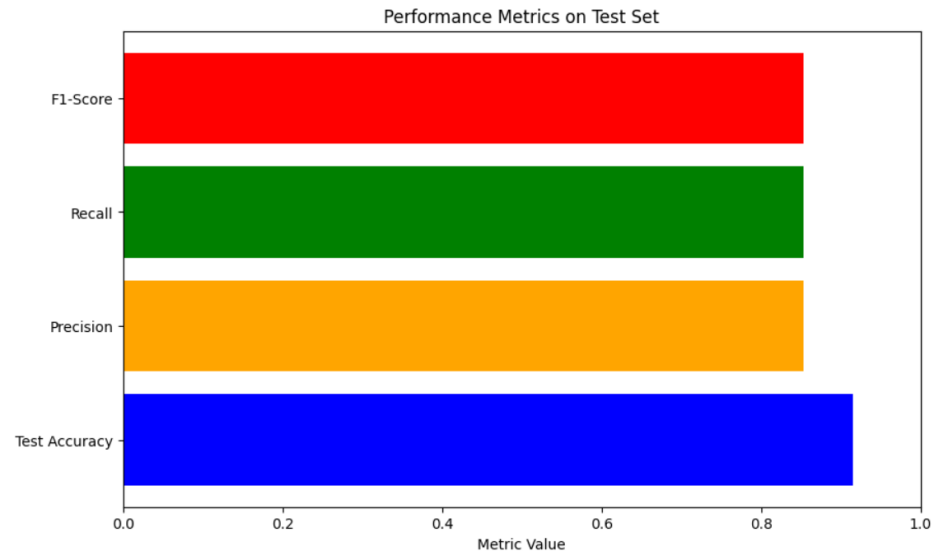
Average(CV) Accuracy: 0.8621  
Accuracy on Test: 0.8291  
Precision: 0.6458  
Recall: 0.9118  
F1-score: 0.7561

Đánh giá:

* CV > Test và |0.8621-0.8291| < 0.1 => Mô hình không bị Overfitting.
* Khoảng cách chênh lệch CV qua từng fold nhỏ, không ảnh hưởng lớn => Mô hình tiếp thu rất tốt.
* Recall ổn chứng tỏ mô hình phân loại được nhiều lớp 1, Precision không được cao trong việc chẩn đoán chính xác nhiều lớp 1, F1 tóm lại được độ hiệu quả, hiệu suất khá ổn của mô hình, nhưng vẫn tiếp tục thua so với RandomForest.

**CatBoostClassifier:**





*Hình 6.15. Chỉ tiêu của mô hình cải thiện CatBoost*

Kết quả đánh giá của mô hình:

* Average(CV) Accuracy: 0.9318
* Accuracy on Test: 0.9145
* Precision: 0.8529
* Recall: 0.8529
* F1-score: 0.8529

Đánh giá:

* CV > Test và |0.9318-0.9145| < 0.1 => Mô hình không bị Overfitting.
* Khoảng cách chênh lệch CV qua từng fold nhỏ, không ảnh hưởng lớn => Mô hình tiếp thu rất tốt. Nhưng sự chính xác qua từng fold giảm xuống khá nhiều dẫn đến sự quan ngại về hiệu suất với mô hình (không ổn định).
* Recall, Precision đều ổn định và cân bằng trong việc phân loại và chẩn đoán chính xác nhiều lớp 1, F1 tóm lại được độ hiệu quả, hiệu suất tốt của mô hình.

**6.4. So sánh kết quả chẩn đoán**

Lựa chọn 2 mô hình có hiệu suất tốt là RandomForest và CatBoost để chẩn đoán mẫu và đưa ra nhận xét, đánh giá.

Vì không có dữ liệu ngoài, nên sẽ sử dụng chính tập dữ liệu thô qua tiền xử lý như là tập chẩn đoán để kiểm tra, đánh giá khả năng chẩn đoán của mô hình.

Tập chẩn đoán này cần được trải qua tiền xử lý tương tự với tiền xử lý dữ liệu huấn luyện của mô hình. Áp dụng giá trị tính toán với số liệu của tập chẩn đoán, không phải tập huấn luyện hay tập kiểm tra vừa rồi.

**Trực quan hóa kết quả chẩn đoán:**

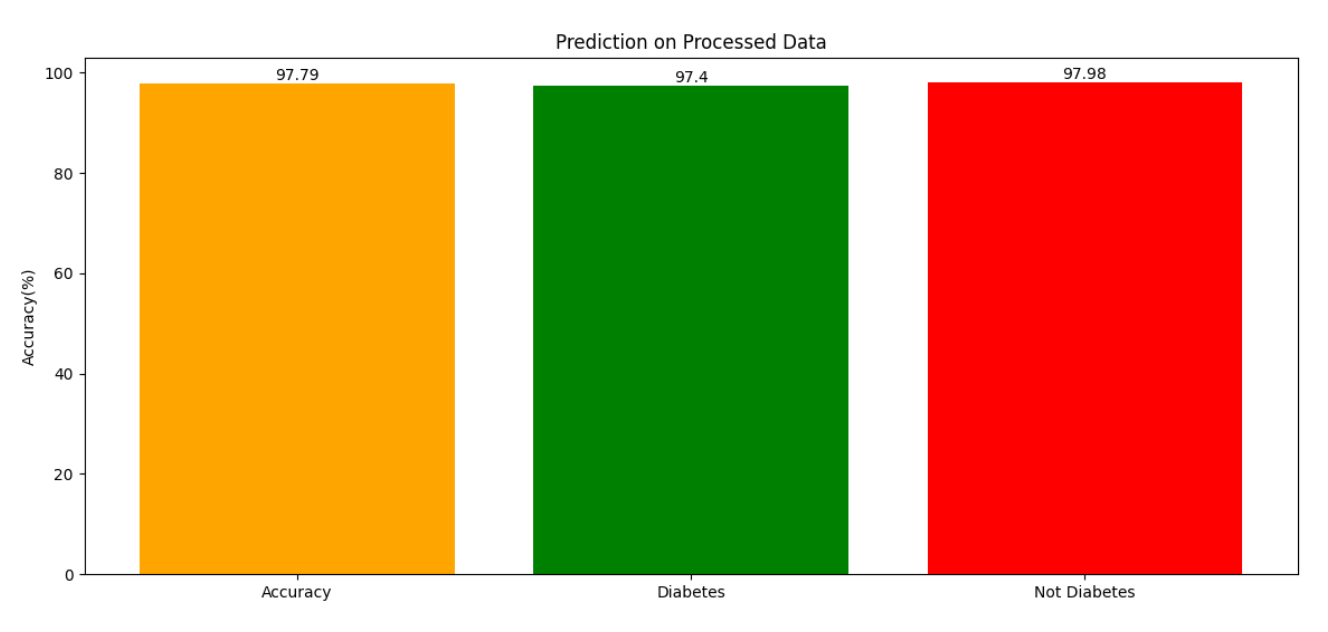
|  |
| --- |
| y\_proba = rf.predict\_proba(df\_x)[:, 1] y\_pred = (y\_proba >= 0.6).astype(int)  df[**'Prediction'**] = y\_pred print(df) accuracy = (((df[**'Prediction'**] == df[**'Outcome'**]).sum()/df.shape[0])\*100).round(2) accuracy\_1 = ((((df[**'Prediction'**]==1) & (df[**'Outcome'**]==1)).sum()/(df[**'Outcome'**]==1).sum())\*100).round(2) accuracy\_0 = ((((df[**'Prediction'**]==0) & (df[**'Outcome'**]==0)).sum()/(df[**'Outcome'**]==0).sum())\*100).round(2)  *# Accuracy is float, need to put it in [] instead of keys(), values()* plt.figure(figsize=(14,6)) bar = plt.bar([**'Accuracy'**],[accuracy],color=**'orange'**) plt.bar\_label(bar) bar\_1 = plt.bar([**'Diabetes'**],[accuracy\_1],color=**'green'**) plt.bar\_label(bar\_1) bar\_0 = plt.bar([**'Not Diabetes'**],[accuracy\_0],color=**'red'**) plt.bar\_label(bar\_0) plt.title(**'Prediction on Processed Data'**) plt.ylabel(**'Accuracy(%)'**) plt.show() |

Tạo chẩn đoán mẫu với ngưỡng tương ứng với ngưỡng của mô hình.

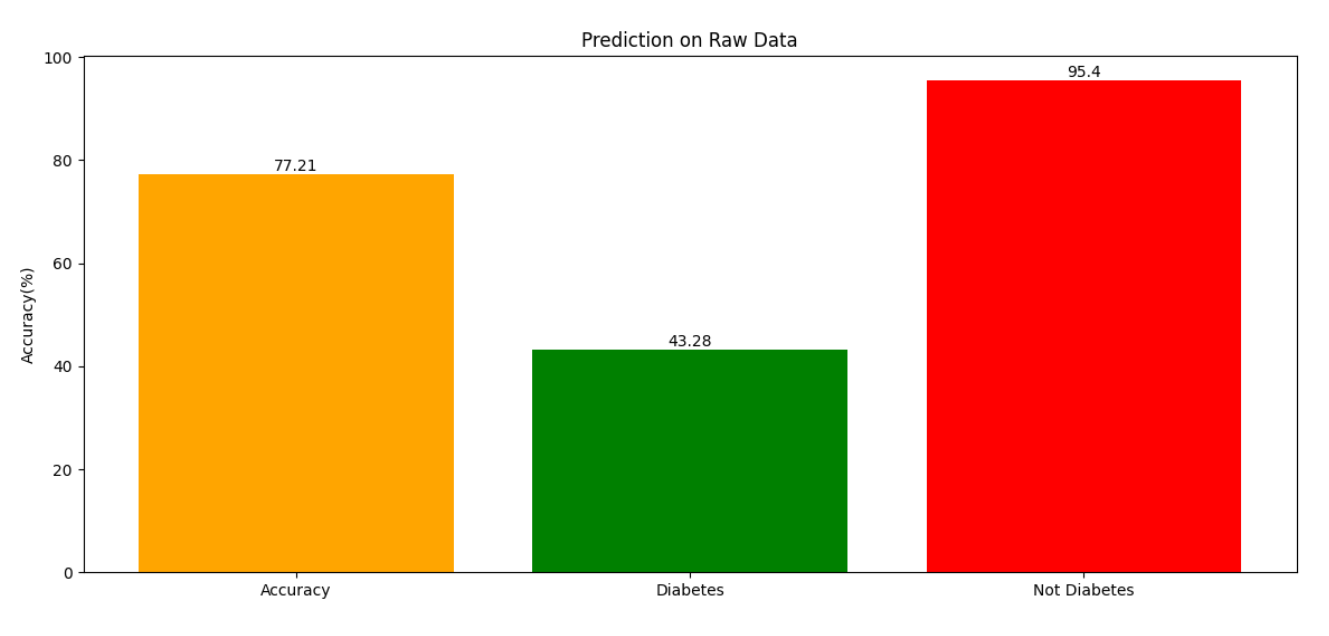
Gán kết quả chẩn đoán mẫu y\_pred vào cột cuối cùng của bảng với tên cột là ‘Prediction’.

* accuracy: là số lượng % độ chính xác chẩn đoán đúng với thực tế trên tập mẫu:
* accuracy\_1: là số lượng % độ chính xác chẩn đoán đúng với thực tế trên lớp 1 trên tập mẫu:
* accuracy\_0: là số lượng % độ chính xác chẩn đoán đúng với thực tế trên của lớp 0 trên tập mẫu:
* Kết quả sẽ được làm tròn 2 thập phân.

**RandomForestClassifier:**



*Hình 6.16. chẩn đoán mẫu đã qua tiền xử lý của RandomForest*

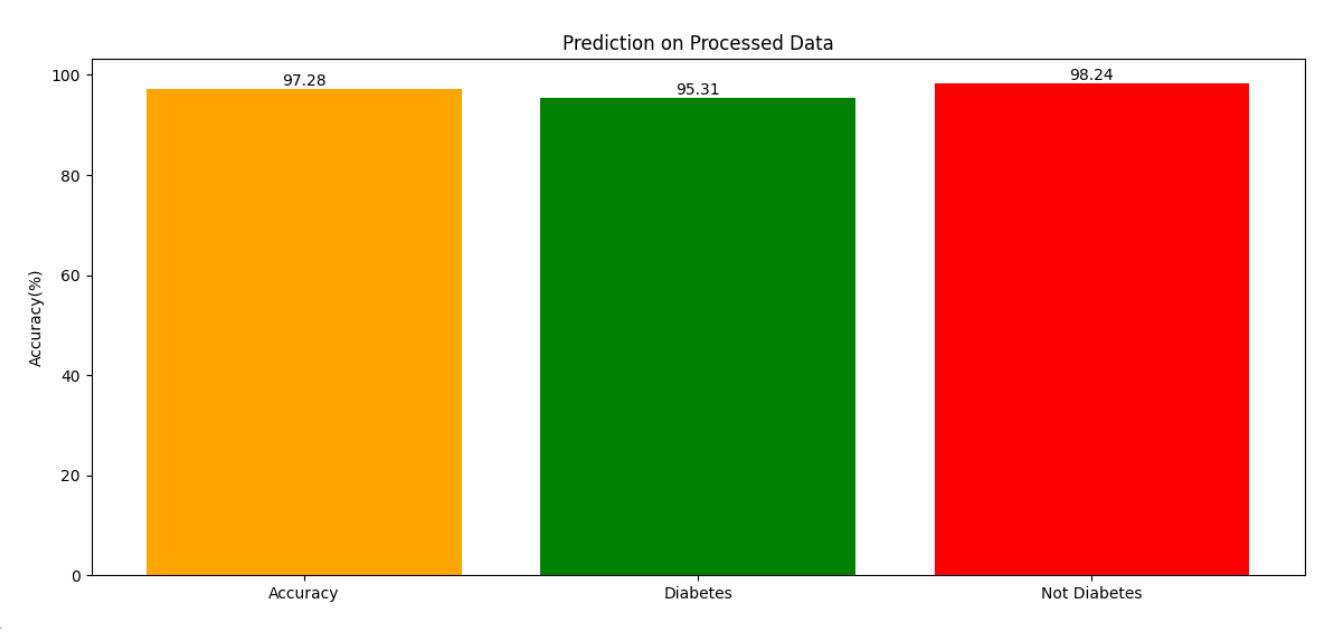


*Hình 6.17. chẩn đoán mẫu thô của RandomForest*

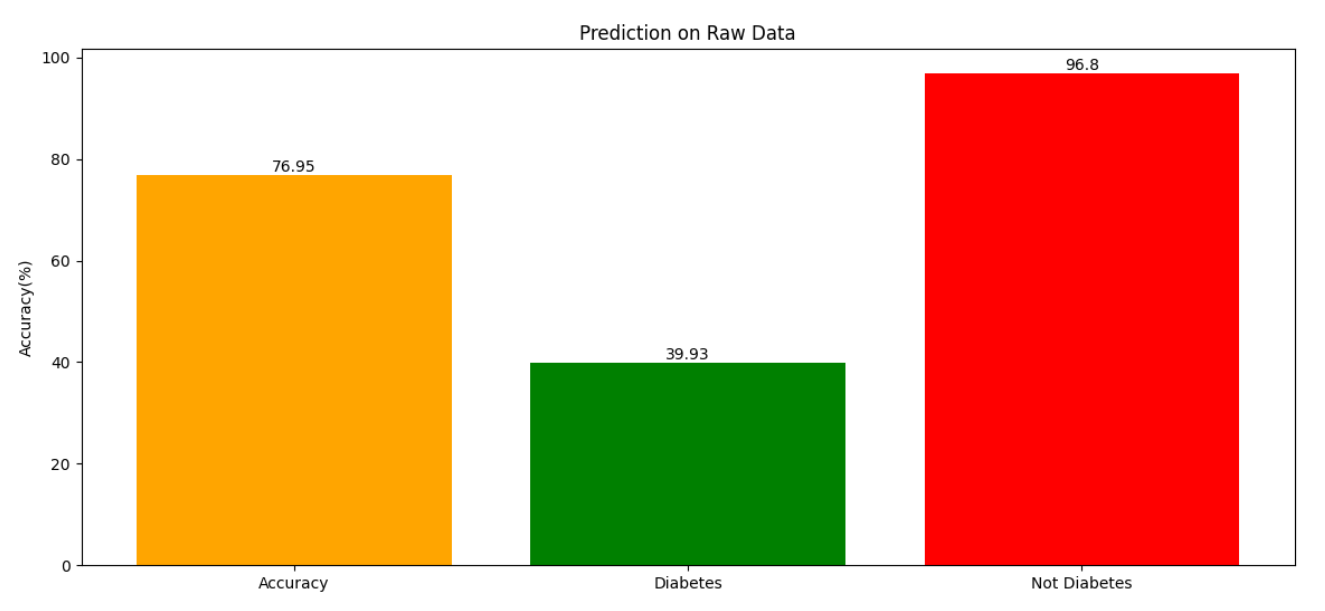
Mô hình RF chẩn đoán mẫu Processed rất tốt lớp 0 (97,4%) và lớp 1 (97,98%), phân biệt được lớp 1 rất hiệu quả mà không có sự chênh lệch với lớp 0.

Với dữ liệu thô, mẫu được chẩn đoán hầu hết là lớp 0, không phân biệt được lớp 1, đây là một dấu hiệu tốt chứng minh mô hình không bị Overfitting hay bị rò rĩ dữ liệu.

**CatBoostClassifier:**



*Hình 6.18. chẩn đoán mẫu đã qua tiền xử lý của CatBoost*



*Hình 6.19. chẩn đoán mẫu thô của CatBoost*

Mô hình CB chẩn đoán mẫu Processed rất tốt lớp 0 (95.31%) và lớp 1 (98,24%), phân biệt 2 lớp tốt. Nhưng mô hình chẩn đoán đúng lớp 0 nhiều hơn và có sự chênh lệch khá lớn với lớp 1.

Với dữ liệu thô, mẫu được chẩn đoán hầu hết là lớp 0, không phân biệt được lớp 1, đây là một dấu hiệu tốt chứng minh mô hình không bị Overfitting hay bị rò rĩ dữ liệu

**Nhận xét:**

* Cả 2 mô hình đều được huấn luyện và dữ liệu được xử lý đúng hướng, thể hiện kết quả rất tốt, đánh giá lẫn chẩn đoán đều đạt kết quả mong đợi.
* Mô hình RandomForestClassifier phân biệt lớp 1 và lớp 0 ổn định và cân bằng hơn mô hình CatBoostClassifier, mặc dù mô hình CatBoost nhận diện tốt ở lớp 0, nhưng mục đích của mô hình này, bài toán, hướng đi và dự án là phân biệt 2 lớp ổn định nhất có thể đặc biệt chú trọng lớp 1.

=> Mô hình RandomForest là mô hình có hiệu suất và độ ổn định tốt nhất

**Chương 7: Cài đặt và sử dụng**

**7.1. Cấu trúc dự án**

|  |
| --- |
| ML\_DiabetesPrediction  │ lib\_list.txt  │ Makefile  │ README.md  │ requirements.txt  │  ├───app  │ API.py  │  ├───config  │ dataset\_config.yaml  │ model\_select.yaml  │ preprocess\_config.yaml  │  ├───data  │ ├───processed  │ │ diabetes.csv  │ │ test.csv  │ │ train.csv  │ │  │ └───raw  │ diabetes.csv  │  ├───models  │ CatBoost  │ Logistic  │ RandomForest  │  ├───notebooks  │ Diabetes\_Exploring.ipynb  │ Diabetes\_Inference.ipynb  │ Diabetes\_Modeling.ipynb  │  ├───results  │ ├───figures  │ │ Báo cáo phân loại - CatBoost.png  │ │ Báo cáo phân loại - Logistic.png  │ │ Báo cáo phân loại - RandomForest.png  │ │ Ma trận nhầm lẫn - CatBoost.png  │ │ Ma trận nhầm lẫn - Logistic.png  │ │ Ma trận nhầm lẫn - RandomForest.png  │ │  │ └───reports  │ Kết quả mô hình CatBoost.csv  │ Kết quả mô hình Logistic.csv  │ Kết quả mô hình RandomForest.csv  │ PredvsData\_CatBoost.csv  │ PredvsData\_Logistic.csv  │ PredvsData\_RandomForest.csv  │  └───src  │ features\_engineering.py  │ utilities.py  │  ├───dataset  │ download\_data.py  │  ├───evaluate  │ evaluate.py  │  ├───models  │ CatBoost.py  │ Logistic.py  │ RandomForest.py  │  ├───prediction  │ prediction.py  │  ├───preprocessing  │ │ data\_preprocessing.py  │ │  │ ├───balance  │ │ RUS.py  │ │ SMOTE.py  │ │  │ ├───imputation  │ │ KNN.py  │ │ mean.py  │ │ median.py  │ │  │ ├───outliers  │ │ attribute\_wise.py  │ │ row\_wise.py  │ │  │ └───scaler  │ minmax.py  │ robust.py  │ standard.py  │  └───visualization  visualization.py |

Dưới đây là mô tả về cấu trúc của dự án:

**Makefile:** Chứa các lệnh tự động hóa cho việc cài đặt, chạy, hoặc kiểm tra dự án.

**README.md:** Mô tả tổng quan về dự án, cách cài đặt và sử dụng.

**requirements.txt:** Danh sách các thư viện bao gồm các mô đun đi kèm với thư viện để chạy dự án.

**lib\_list.txt:** Tương tự với requirements.txt nhưng là danh sách ngắn hơn không bao gồm mô đun đi kèm mà chỉ chứa các tên thư viện quan trọng.

**app:** Thư mục để chứa ứng dụng  
**-API.py:** Tạo REST API với FastAPI để thực hiện chẩn đoán.\

**config:** Thư mục để chứa các thiết lập  
**-dataset\_config.yaml:** Chọn tên tập dữ liệu để xử lý, đường dẫn tải tập dữ liệu từ Kaggle.  
**-model\_select.yaml:** Chọn mô hình để chạy hoặc tổ hợp mô hình, ngưỡng chẩn đoán của mô hình.  
**-preprocess\_config.yaml:** Thiết lập quy trình xử lý dữ liệu.

**data:** Thư mục để chứa các tập dữ liệu  
**-external:** Chứa dữ liệu từ nguồn bên ngoài, tên tập dữ liệu phải là external.csv.  
**-processed:** Dữ liệu đã qua xử lý sẵn sàng để huấn luyện, chứa train.csv, test.csv, và diabetes.csv.  
**-raw:** Dữ liệu gốc chưa qua xử lý.

**models:** Thư mục để chứa các mô hình đã lưu

**notebooks:** Thư mục để chứa notebook

**results:** Thư mục chứa kết quả mô hình  
**-figures:** Lưu trữ các biểu đồ và ma trận nhầm lẫn của từng mô hình.  
**-reports:** Báo cáo chi tiết về kết quả chẩn đoán, các báo cáo phân loại và so sánh giữa chẩn đoán với thực tế.

**src:** Thư mục để chứa các mã nguồn xử lý, huấn luyện và đánh giá mô hình  
**-features\_engineering.py:** Xử lý kỹ thuật đặc trưng.  
**-utilities.py:** Chứa các hàm tiện ích dùng chung trong dự án.  
**-dataset:** download\_data.py - Tải dữ liệu thô.  
**-evaluate:** evaluate.py - Đánh giá hiệu suất mô hình.  
**-models:** Tệp chứa các mã nguồn huấn luyện và lưu mô hình.  
**-prediction:** prediction.py - chẩn đoán đầu ra của mô hình.

**-preprocessing:** Tiền xử lý dữ liệu  
**--data\_preprocessing.py:** Xử lý dữ liệu.  
**--balance:** Các phương pháp cân bằng dữ liệu như RUS (Random Under Sampling) và SMOTE.  
**--imputation:** Các phương pháp xử lý giá trị thiếu, bao gồm KNN, mean, và median.  
**--outliers:** Các phương pháp phát hiện và xử lý ngoại lai, attribute\_wise.py và row\_wise.py.  
**--scaler:** Các phương pháp chuẩn hóa dữ liệu, bao gồm MinMax, Robust, và Standard scaler.  
**--visualization:** visualization.py - Mã nguồn thực hiện vẽ hình, biểu đồ, ma trận kết quả của mô hình.

**7.2. Cài đặt**

**Tải ngôn ngữ Python:** <https://www.python.org/downloads/>

Sử dụng một môi trường ảo:  
**Tải Pycharm:** <https://www.jetbrains.com/pycharm/download>

**Cài đặt các thư viện cần thiết trong terminal của môi trường ảo:**

Có 2 file yêu cầu:

* requirements.txt là toàn bộ thư viện bao gồm module nhỏ được tương thích phiên bản với nhau.
* lib\_list là danh sách từng tên của thư viện.

Sử dụng 1 trong 2 cách tải thư viện ở dưới đây:

* Xem danh sách trong lib\_list.txt và cài đặt từng thư viện với lệnh:

|  |
| --- |
| pip install (tên thư viện) |

* Sử dụng lệnh này với requirements.txt để tải hết toàn bộ cùng một lúc:

|  |
| --- |
| pip install -r requirements.txt |

**7.3. Sử dụng**

Sử dụng lệnh python thông qua terminal ảo hoặc chạy trực tiếp file python để chạy chương trình.

* Chọn cách xử lý dữ liệu trong **preprocess\_config.yaml.**
* Trước khi chạy, hãy chọn mô hình sử dụng và ngưỡng chẩn đoán của mô hình trong **model\_select.yaml** để đánh giá (evaluate.py), vẽ biểu đồ trực quan (visualization.py), chẩn đoán kết quả (prediction.py).

Đảm bảo mọi thiết lập đều được cài đặt đầy đủ. Mọi thiết lập đều nằm trong folder **config**.

Với cách xử lý dữ liệu qua **Chương 4** cùng với mục **6.3. Cải thiện mô hình,** tạo ra mô hình có hiệu suất tốt nhất đã được đưa ra ở mục **6.4. So sánh kết quả chẩn đoán**, đó chính là RandomForest. Thiết lập sẽ được cài đặt như sau:

* preprocess\_config.yaml:

|  |
| --- |
| scaling: "disable" *# Any input except "enable" will disable it* scaler: "robust" *# Mustn't be blanked if scaling is enable* imputation: "median" *# Mustn't be blanked* balance\_method: "SMOTE" *# None to not use any method* outliers\_method: "attribute\_wise" *# None to not use any method* |

Các tên phương pháp dựa trên tên file.py trong folder bước xử lý trong folder tiền xử lý. Cụ thể là src.preprocess.(bước xử lý).(tên phương pháp).

Ví dụ: “src.preprocess.balance.SMOTE” sẽ tương ứng với balance\_method: “SMOTE”.

* model\_select.yaml:

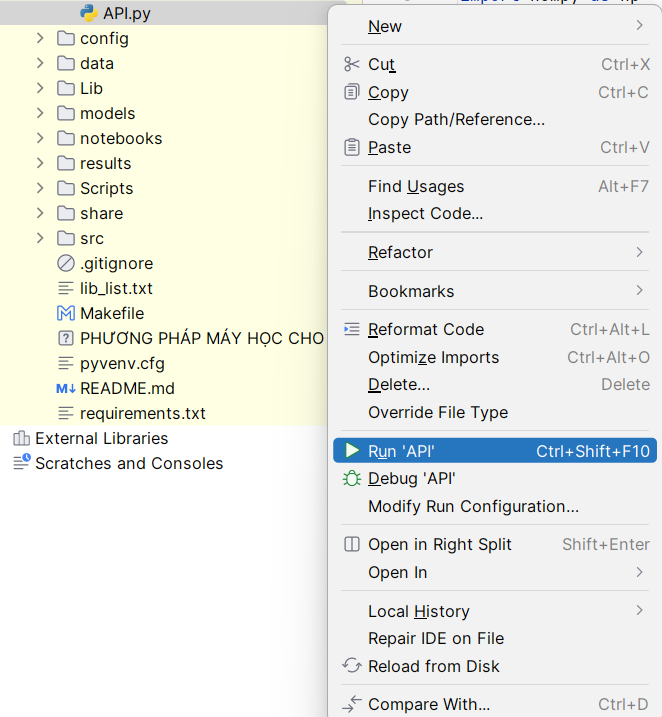
|  |
| --- |
| *# Predict Visualize Evaluate API* model:  name: "RandomForest"  threshold: "0.6" |

Chọn mô hình có hiệu suất tốt nhất, RandomForest.

Ngưỡng chẩn đoán của mô hình là 0.6 (tốt nhất).

* **Chạy dự án:**

Trong Pycharm, ấn chuột phải lên file.py và chọn Run.



*Hình 7.1: Chạy chương trình*

* **Tải tập dữ liệu thô:** src\dataset\download.py
* **Tiền xử lý dữ liệu:** src\preprocess\data\_preprocessing.py.
* **Huấn luyện:** src\models\RandomForest.py
* **Đánh giá mô hình:** src\evaluate\evaluate.py
* **Biểu đồ trực quan:** src\visualization\visualization.py
* **chẩn đoán kết quả:** src\prediction\prediction.py
* **API:** src\app\API.py (sẽ được giải thích thêm ở mục **7.4. Kiểm thử API**)

**7.4. Kiểm thử API**

**a) Khái niệm**

API (Application Programming Interface - Giao diện lập trình ứng dụng) là một tập hợp các quy tắc và giao thức cho phép các ứng dụng phần mềm khác nhau giao tiếp với nhau.

**Chức năng chính của API**:

* + Cung cấp cách truy cập và thao tác dữ liệu hoặc tính năng của một ứng dụng từ các ứng dụng khác.
  + Kết nối các thành phần, như ứng dụng client (frontend) và server (backend).

**Đặc điểm của API**:

* + **Phạm vi ứng dụng rộng**: API có thể dùng để giao tiếp giữa các hệ thống khác nhau như máy chủ, cơ sở dữ liệu hoặc dịch vụ web.
  + **Hình thức**: API có thể là RESTful API (phổ biến với HTTP), SOAP API, hoặc GraphQL.
  + **Tiện lợi**: Hầu hết các API đều cung cấp một giao diện dễ sử dụng với các endpoint để thực hiện các hành động cụ thể, chẳng hạn như đọc, ghi, hoặc cập nhật dữ liệu.

**Ví dụ về API**:

* + Facebook cung cấp API để các nhà phát triển tích hợp hệ thống của họ với Facebook (như đăng bài viết từ ứng dụng).
  + Google Maps API cho phép nhúng bản đồ Google Maps vào một website.

**b) FastAPI**

**FastAPI** là một framework web hiện đại và nhanh dành cho Python, được thiết kế để xây dựng các ứng dụng API, phù hợp để xây dựng các API machine learning.

**Tính năng chính của FastAPI**:

* 1. **Nhanh chóng:**
* Tốc độ xử lý cao nhờ tận dụng tính năng không đồng bộ (asynchronous).
* Tương đương với các framework hiệu suất cao như Node.js hoặc Go.
  1. **Hỗ trợ tự động**:
* FastAPI tự động kiểm tra dữ liệu đầu vào dựa trên mô hình khai báo (schema) thông qua **Pydantic**.
* Tự động sinh **tài liệu API** bằng Swagger UI hoặc ReDoc mà không cần cấu hình.
  1. **Thân thiện với lập trình viên**:
* Dễ sử dụng và hỗ trợ tốt gợi ý mã (type hints), giúp phát triển nhanh hơn với ít lỗi hơn.
  1. **Tích hợp dễ dàng với Machine Learning**:
* FastAPI lý tưởng để triển khai các dự án Machine Learning hoặc Data Science do tính tương thích với các thư viện như scikit-learn, TensorFlow, PyTorch.

**Các thành phần chính**:

**Router (định tuyến)**: Xác định các **endpoint** để API phục vụ (ví dụ: /predict).

**Middleware**: Thêm các tầng xử lý trước hoặc sau yêu cầu (như xử lý bảo mật, quản lý session, CORS).

**Dependency Injection**: Hỗ trợ xây dựng mã linh hoạt và dễ kiểm thử.

Trong ứng dụng này, FastAPI được dùng để triển khai endpoint /predict/ để:

* Tiếp nhận dữ liệu sức khỏe từ phía người dùng.
* Xử lý dữ liệu qua pipeline của mô hình.
* Trả về kết quả "Diabetes" (có nguy cơ mắc bệnh) hoặc "Not Diabetes" (không có nguy cơ).

**c) Mục tiêu**

**Chức năng**: Đảm bảo rằng:

* /predict/ xử lý đầu vào chính xác và trả về kết quả chẩn đoán hợp lệ.
* Xử lý các trường hợp dữ liệu thiếu hoặc sai định dạng một cách hợp lý.

**Hiệu năng**:

* Kiểm tra thời gian phản hồi khi nhận dữ liệu từ client.
* Đảm bảo hệ thống hoạt động tốt với các yêu cầu đồng thời.

**Khả năng xử lý lỗi**:

* Phát hiện và xử lý lỗi (nếu dữ liệu đầu vào không đúng hoặc xảy ra vấn đề với mô hình).
* Đảm bảo API luôn trả về phản hồi (ngay cả khi có lỗi).

**d) Xây dựng và kiểm thử**

**d.1) OOP:**

|  |
| --- |
| class Diabetes(BaseModel):  Pregnancies: int  Glucose: int  BloodPressure: int  SkinThickness: int  Insulin: float  BMI: float  DiabetesPedigreeFunction: float  Age: int |

**d.2) Khởi tạo FastAPI:** app = FastAPI()

**d.3) Tích hợp mô hình, chuẩn bị dữ liệu:**

**Chuẩn bị và tích hợp mô hình:**

|  |
| --- |
| *# Load YAML config* name = utilities.model\_select()['name'] threshold = float(utilities.model\_select()['threshold']) *# Check whether the model exists or not* check = bool(utilities.check\_model(name)) if check is False:  return None full\_pipeline = utilities.joblib\_load(f'{name}') |

* Đảm bảo API sử dụng đúng mô hình chẩn đoán.
* Ngưỡng chẩn đoán (threshold) cho phép điều chỉnh tùy vào mục tiêu.

**Tạo giới hạn đặc trưng:**

Để không chẩn đoán đặc trưng nếu nó không hợp với số liệu thực tế:

|  |
| --- |
| *# Define feature limits* feature\_limits = {  "Pregnancies": (0, 20),  "Glucose": (50, 300),  "BloodPressure": (30, 200),  "SkinThickness": (0, 99),  "Insulin": (0, 600),  "BMI": (10, 70),  "DiabetesPedigreeFunction": (0, 2),  "Age": (1, 120), }  def validate\_limits(data: dict) -> str:  *"""Check if input data is within the defined limits."""* for feature, (lower, upper) in feature\_limits.items():  if data[feature] < lower or data[feature] > upper:  return f"Dữ liệu không thực tế {feature}: {data[feature]}"  return "valid" |

**d.4) Định nghĩa endpoint:**

**Endpoint chính:** /predict

* Sử dụng phương thức **POST** để nhận dữ liệu đầu vào dưới dạng JSON.
* Dữ liệu đầu vào được chuyển đổi thành DataFrame (phù hợp với đầu vào của mô hình học máy).
* Mô hình chẩn đoán trả về xác suất, so sánh với ngưỡng để đưa ra kết quả cuối cùng.

|  |
| --- |
| @app.post('/predict/') async def predict\_diabetes(input\_data: Diabetes):  try:  *# Prepare input data as a dictionary* input\_dict = {  'Pregnancies': int(input\_data.Pregnancies),  'Glucose': int(input\_data.Glucose),  'BloodPressure': int(input\_data.BloodPressure),  'SkinThickness': int(input\_data.SkinThickness),  'Insulin': float(input\_data.Insulin),  'BMI': float(input\_data.BMI),  'DiabetesPedigreeFunction': float(input\_data.DiabetesPedigreeFunction),  'Age': int(input\_data.Age)  }   *# Validate input data against feature limits* validation\_result = validate\_limits(input\_dict)  if validation\_result != "valid":  return {"prediction": validation\_result}   *# Convert input data to a DataFrame* x\_values = pd.DataFrame([input\_dict])   *# Make prediction* prediction\_proba = full\_pipeline.predict\_proba(x\_values)[:, 1]  prediction = (prediction\_proba >= threshold).astype(int)  *# prediction = full\_pipeline.predict(x\_values)* prediction\_result = 'Diabetes' if prediction[0] == 1 else 'Not Diabetes'  return {'prediction': prediction\_result}   except Exception as e:  raise HTTPException(status\_code=400, detail="API server is closed")  return app |

**Endpoint chào mừng:** /

* Mục đích kiểm tra nhanh API đang hoạt động.

|  |
| --- |
| @app.get('/') def welcome():  return {"Welcome": "User"} |

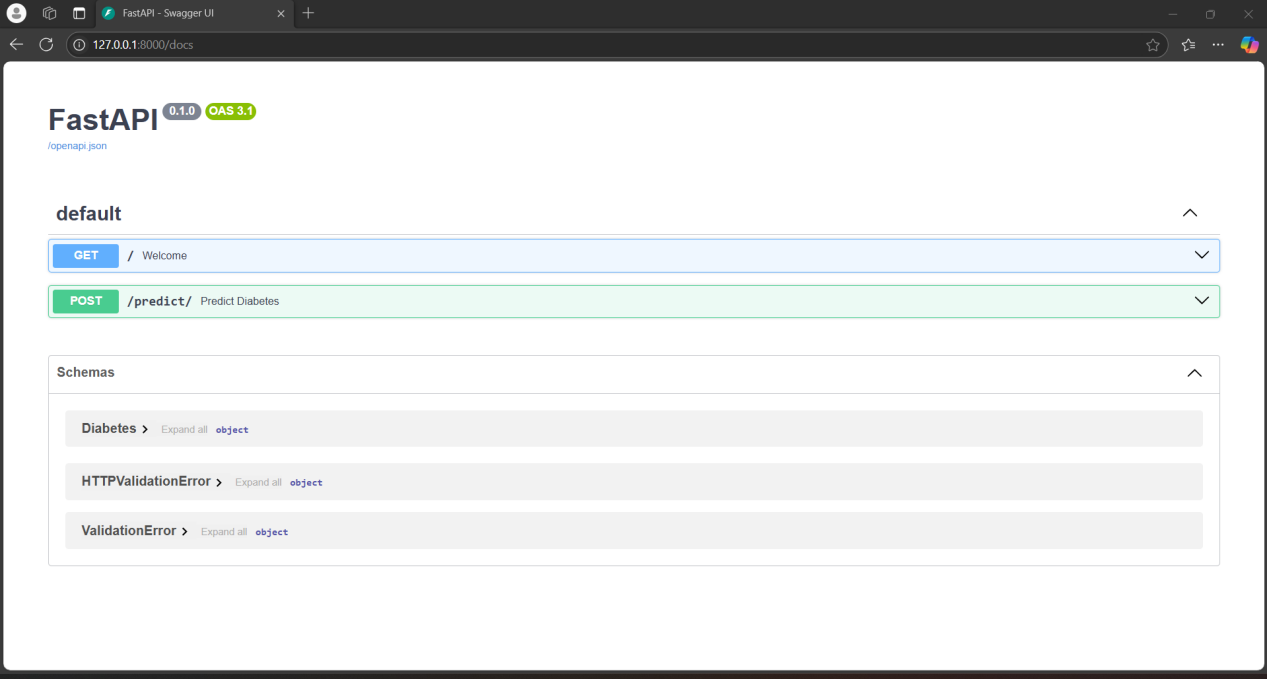
**d.5) Chạy ứng dụng FastAPI trên cổng 8000 bằng uvicorn, sử dụng localhost:** uvicorn.run(app, host='127.0.0.1', port=8000)

**d.6) Kiểm thử**:

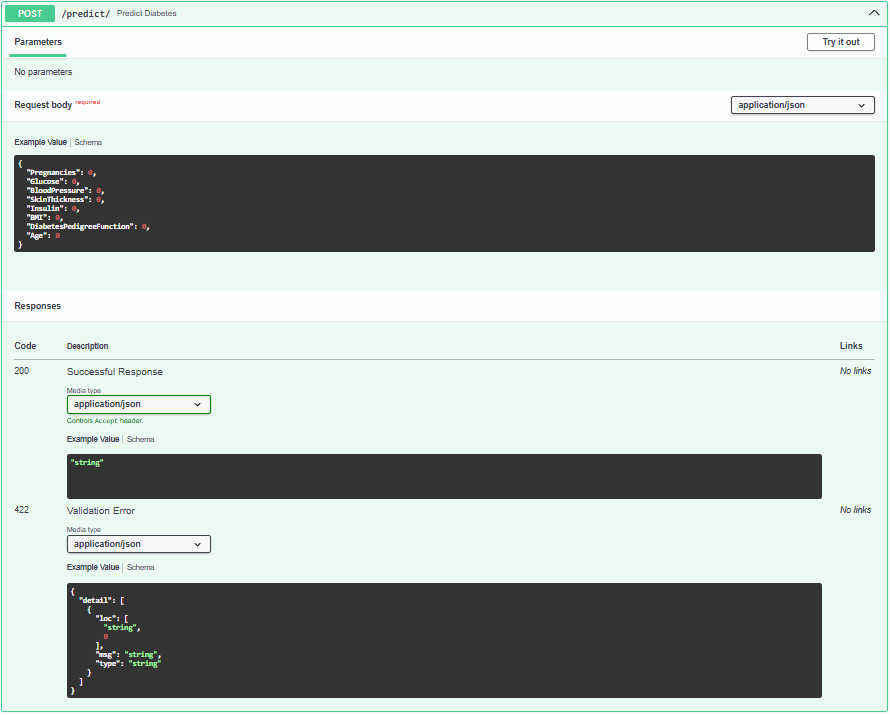
Chạy terminal ảo ở đường dẫn file python (API.py) hiện tại với 1 trong các lệnh:

* uvicorn API:API --reload --factory
* python -m API (chỉ hoạt động nếu chương trình chạy trong \_\_main\_\_)
* python API.py

Nhấn vào liên kết đường dẫn <http://127.0.0.1:8000>, sau đó thêm /docs vào liên kết đường dẫn http, nó sẽ dẫn đến Swagger UI của FastAPI, đây là một trang/công cụ hỗ trợ các endpoint cũng như hỗ trợ chẩn đoán bệnh tiểu đường qua mã nguồn đã xây dựng trước đó:



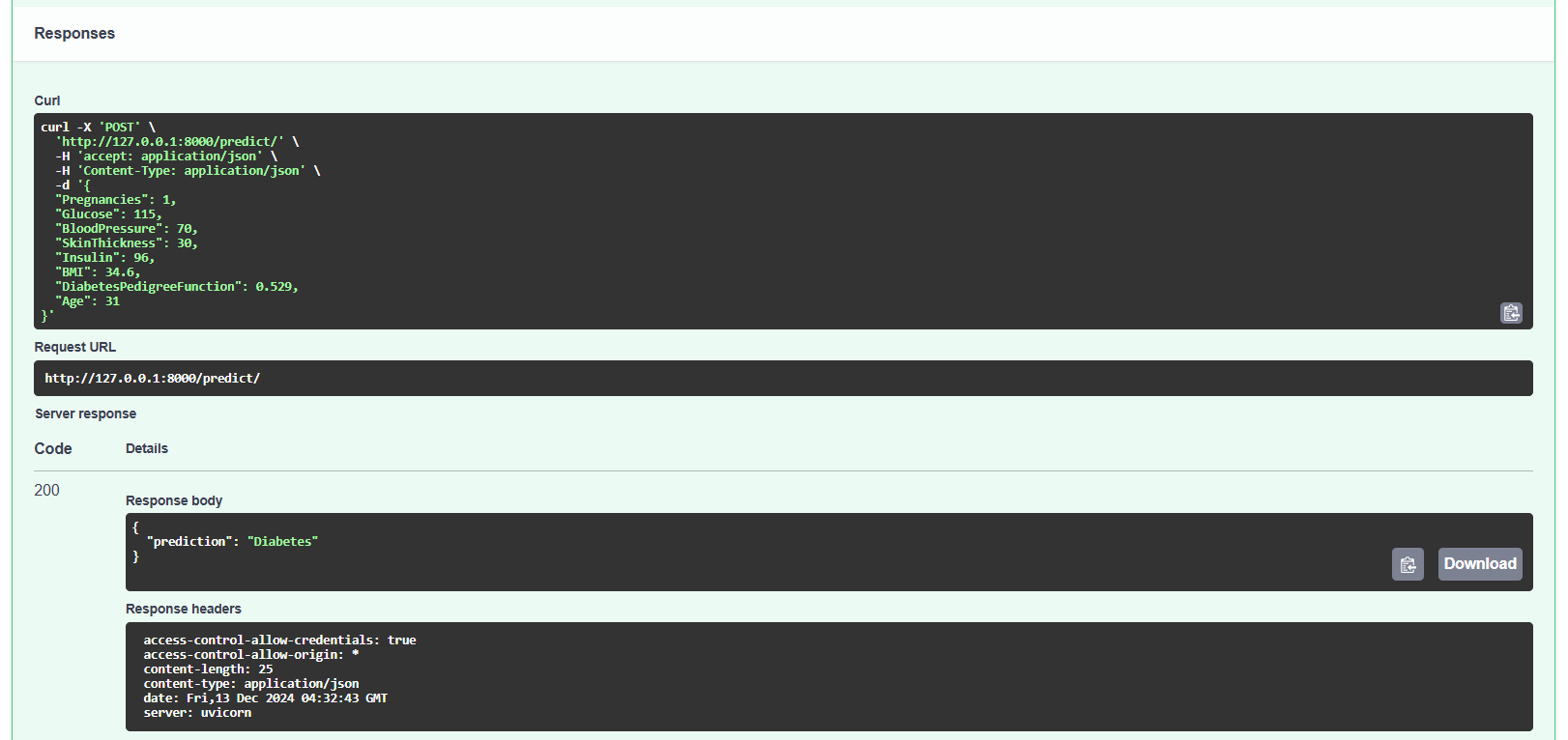
Ấn vào POST /predict/ Predict Diabetes sẽ mở công cụ POST chẩn đoán tiểu đường:



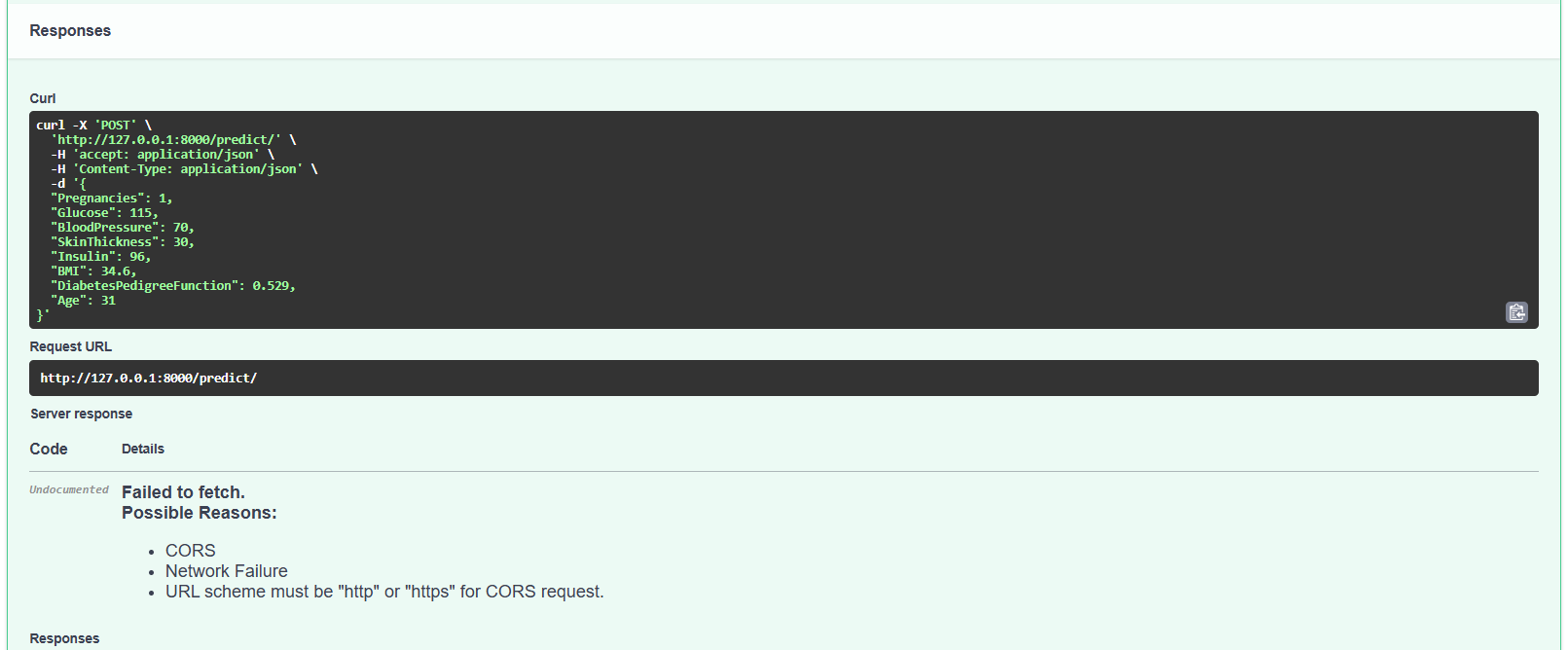
Ấn vào Try It Out để cho phép sử dụng, điền lần lượt các đặc trưng ở phần **Request body**:

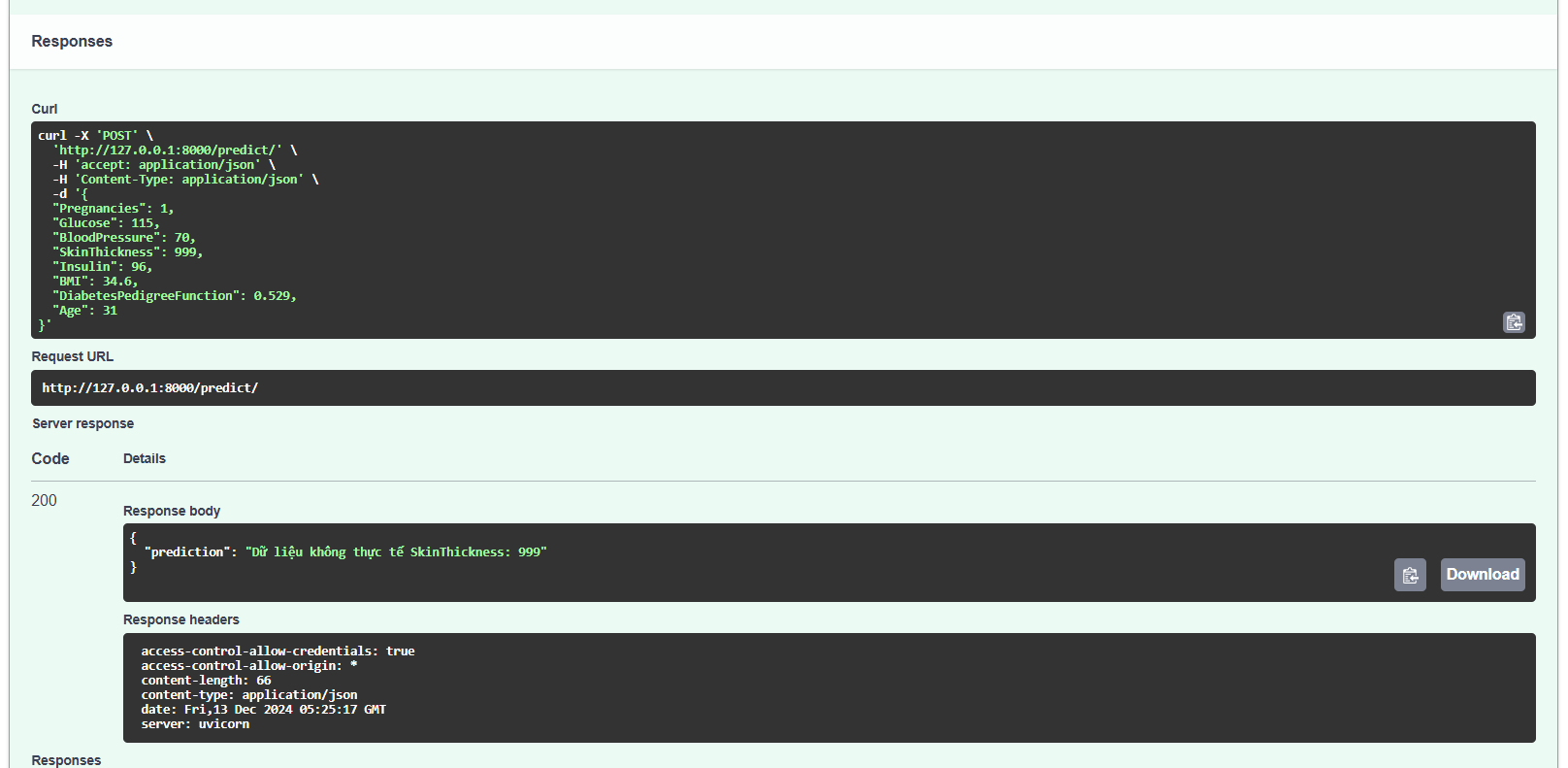
|  |
| --- |
| {  "Pregnancies": 1,  "Glucose": 115,  "BloodPressure": 70,  "SkinThickness": 30,  "Insulin": 96,  "BMI": 34.6,  "DiabetesPedigreeFunction": 0.529,  "Age": 31  } |

Ấn nút xanh dương Execute bên dưới để nhận kết quả tiểu đường ở hộp **Response body** trong phần **Responses**:



**Hậu trường hợp:** Nếu Server chưa được chạy hay chẩn đoán không hợp lí, sẽ xuất hiện các lỗi như sau:





**e) Đánh giá và cải thiện**

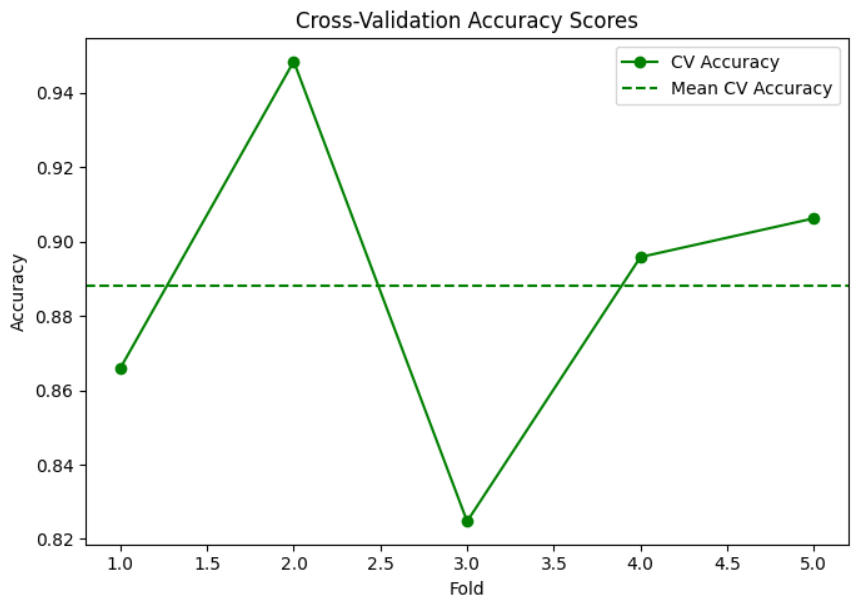
* **Điểm tốt**:
  + Mô hình trả về kết quả nhanh (<500ms).
* **Điểm cần cải thiện**:
  + Xây dựng thêm HTML và kết nối FastAPI với HTML để sử dụng dễ dàng.
  + Xây dựng server để host và public cho nhiều đối tượng người dùng.
  + Tối ưu hóa pipeline nếu thời gian phản hồi tăng khi tải lớn.

**Chương 8: Kết quả**

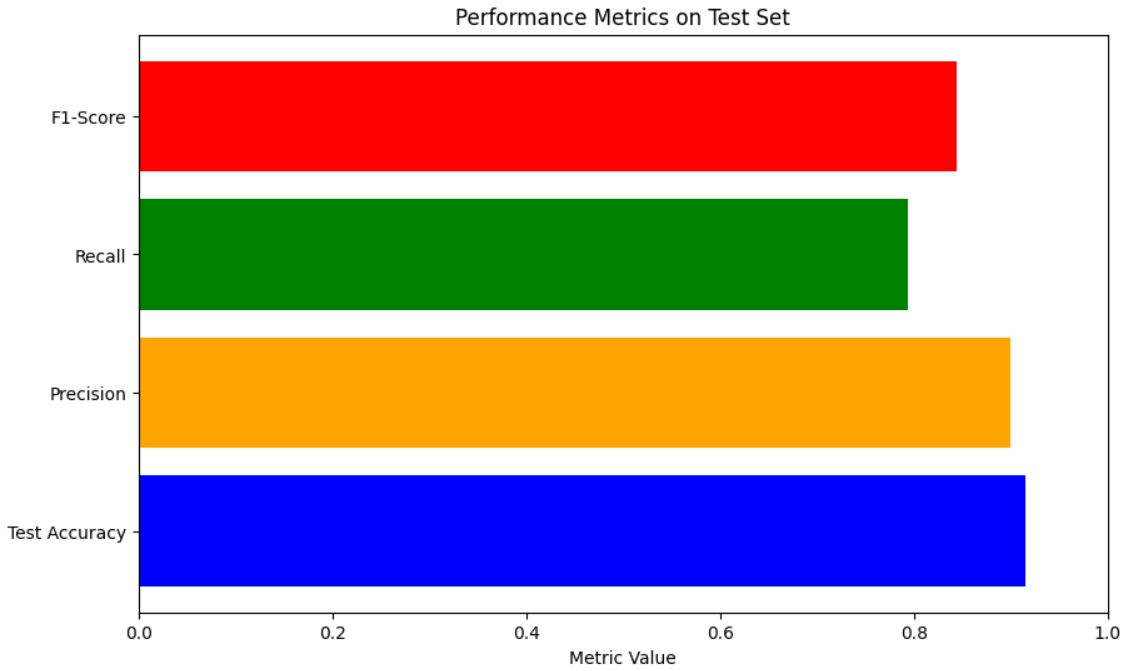
**8.1. Việc chênh lệch nhãn có ảnh hưởng đến mô hình chẩn đoán kết quả hay không?**

Việc chênh lệch nhãn sẽ khiến cho mô hình ít học được lớp cần được phân biệt nhất, cụ thể đó là lớp 1.

Giả sử, với mô hình RandomForest:



*Hình 8.1. Cross-validation của Tập huấn luyện*

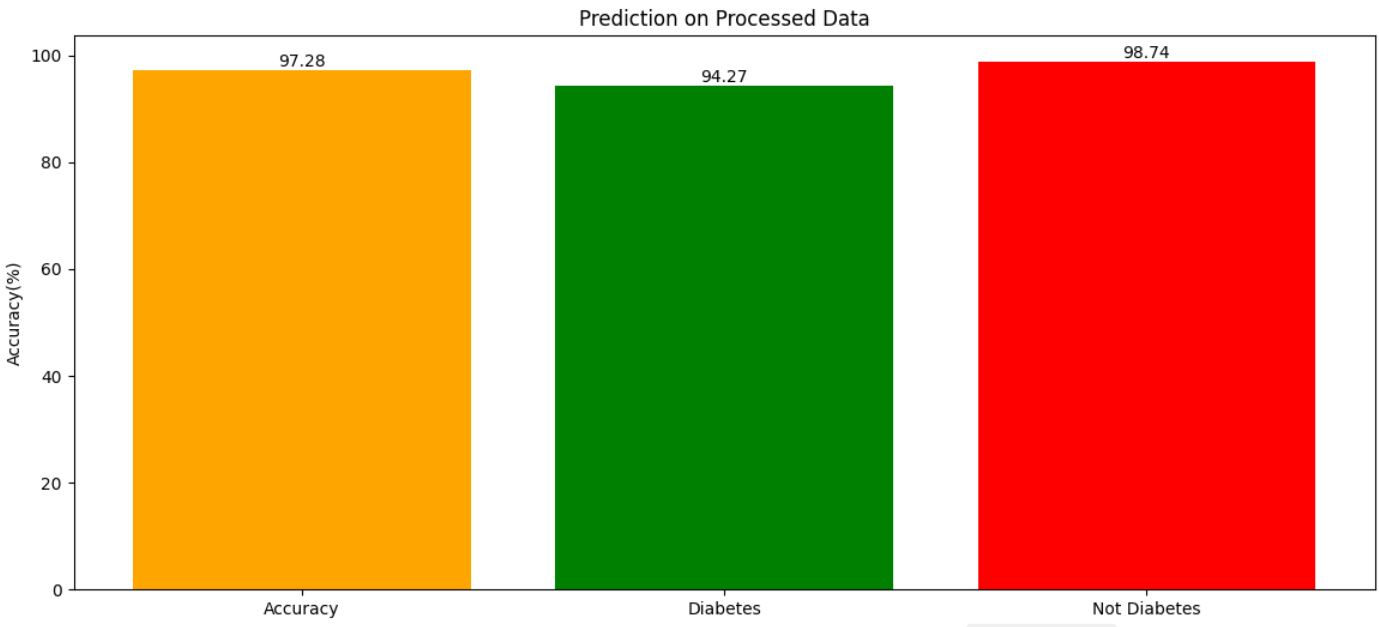


*HÌnh 8.2. Chỉ tiêu đánh giá của Tập kiểm tra*

**Kết quả đánh giá của mô hình:**

* Average(CV) Accuracy: 0.8883
* Accuracy on Test: 0.9145
* Precision: 0.9000
* Recall: 0.7941
* F1-score: 0.8438
* Với dữ liệu chưa cân bằng nhãn, mô hình không học đầy đủ chi tiết đặc điểm lớp 1, khiến hiệu suất mô hình qua xác thực chéo thấp.
* CV < Test và |0.8883-0.9145| < 0.1 => Mô hình không bị Underfitting.
* Precision có độ chính xác cao chứng tỏ mô hình chẩn đoán chính xác rất nhiều lớp 1. Recall không quá cao, có vẻ mô hình vẫn chưa học đủ lớp 1 khiến cho khả năng nhận diện mẫu thực lớp 1 kém. F1 tóm lại được hiệu suất vẫn tốt, nhưng vẫn rất quan ngại với Recall và CV Accuracy, vì điều này biểu hiện mô hình mặc dù có hiệu suất tốt với tập kiểm tra, dữ liệu mới, nhưng tập xác thực hiệu suất không được tốt như tập kiểm tra.

**Đánh giá chẩn đoán:**



*Hình 8.3. chẩn đoán mẫu đã qua tiền xử lý*

Dựa vào kết quả chẩn đoán, quả thực mô hình vẫn chưa phân biệt tốt lớp 1, vì sự chênh lệch của nhãn trong dữ liệu đặc biệt lớp 1 chiếm rất ít, khiến cho mô hình học không đủ chi tiết. Dẫn đến chẩn đoán lớp 1 ít hơn, việc phân loại lớp không cân bằng.

=> Nếu không có sự cân bằng nhãn, điều này sẽ khiến cho mô hình không được đáp ứng trong việc cân bằng phân biệt 2 lớp đặc biệt lớp phải được chú trọng nhất, lớp 1. Biểu hiện qua sự chênh lệch rất lớn trong CV và chỉ tiêu Recall không được cao mặc dù chỉ tiêu F1 và sự chính xác tập kiểm tra cao .

**8.2. Vì sao không loại bỏ hai đặc trưng có mức tương quan thấp với nhãn?**

Việc loại bỏ đặc trưng có mức tương quan thấp với nhãn là điều nên được xem xét qua tập dữ liệu

* Với tập **dữ liệu lớn**, nếu việc bỏ đi đặc trưng có mức tương quan thấp khiến cho mô hình có hiệu suất tốt về mọi mặt, thì có thể bỏ. Ngược lại, thì không nên bỏ.
* Với tập **dữ liệu nhỏ**, việc bỏ đi đặc trưng có mức tương quan thấp là khuyến khích **không nên**, vì đó là những đặc trưng quan trọng và sẽ giúp cho mô hình có thêm những chi tiết đó để học, đặc biệt nó là tập dữ liệu nhỏ.

=> Dữ liệu cho dự án hiện tại đang sử dụng là tập dữ liệu nhỏ, nên việc loại bỏ đặc trưng có mức tương quan thấp với nhãn là không được áp dụng, vì điều này khiến cho mô hình học hỏi ít thông tin hơn, hiệu suất cũng như khả năng chẩn đoán sẽ không như mong muốn.

**Chương 9: Đánh giá và kết luận**

**9.1. Đánh giá kết quả nghiên cứu**

Trong nghiên cứu này, chúng tôi đã triển khai và thử nghiệm các phương pháp học máy để chẩn đoán bệnh tiểu đường dựa trên dữ liệu y tế. Kết quả đạt được thể hiện qua các bước tiền xử lý dữ liệu, tối ưu hóa mô hình và đánh giá hiệu năng trên bộ dữ liệu thử nghiệm.

Mô hình Random Forest đạt độ chính xác tổng thể 97.79% trên tập dữ liệu, với độ chính xác cao trong việc phân loại bệnh nhân không mắc tiểu đường (97.98%) và phân loại bệnh nhân mắc tiểu đường (97.4%). Những kết quả này chứng minh tính hiệu quả của phương pháp xử lý dữ liệu và thiết kế mô hình.

**9.2. Những hạn chế trong nghiên cứu**

Mặc dù đạt được những kết quả tích cực, nghiên cứu vẫn tồn tại một số hạn chế cần lưu ý:

**Dữ liệu hạn chế về kích thước và dữ liệu thiếu:**

Bộ dữ liệu chỉ bao gồm 768 mẫu trong đó, 614 mẫu trong tập huấn luyện và 154 mẫu trong tập kiểm tra (sau tiền xử lý còn 660 mẫu trong tập huấn luyện, 117 mẫu trong tập kiểm tra), dẫn đến việc khó tổng quát hóa mô hình trên các mẫu đối tượng khác nhau.

Các giá trị bị thiếu trong các đặc trưng quan trọng như Insulin và BloodPressure đã được điền, ước tính, nhưng cách tiếp cận này có thể gây ra sai số và không đúng với số liệu của bệnh nhân một cách thực tế.

**Mất cân bằng trong dữ liệu:**

Tập dữ liệu ban đầu có sự chênh lệch giữa hai lớp (mắc bệnh và không mắc bệnh), và mặc dù đã sử dụng SMOTE để cân bằng dữ liệu, việc này không hoàn toàn cải thiện phân phối chẩn đoán lớp 1 một cách hoàn hảo.

**9.3. Hướng nghiên cứu tiếp theo**

Để cải thiện và mở rộng nghiên cứu này, một số định hướng nghiên cứu tiếp theo được đề xuất như sau:

**Mở rộng và làm phong phú dữ liệu:**

Thu thập thêm dữ liệu từ các nguồn khác nhau với các đặc trưng y tế chi tiết hơn và đa dạng về dân số, nhằm cải thiện tính tổng quát của mô hình.

Nghiên cứu cách xử lý missing values dựa trên dữ liệu y học thực tế thay vì ước lượng thống kê đơn thuần như điền trực tiếp mà không sử dụng giá trị trung vị, trung bình,… được ước tính từ số liệu của dữ liệu.

**Thử nghiệm nhiều phương pháp mô hình hóa khác:**

Áp dụng nhiều mô hình nâng cao hơn như Gradient Boosting (XGBoost) hoặc Deep Learning để so sánh hiệu quả với mô hình hiện tại.

Xây dựng các mô hình tổ hợp (ensemble) như Voting hoặc Stacking để tận dụng điểm mạnh của nhiều thuật toán khác nhau.

Thử nghiệm nhiều không gian tham số để giúp mô hình học chi tiết và quan sát dữ liệu tốt hơn.

**Phát triển ứng dụng thực tế:**

Thiết kế hệ thống chẩn đoán bệnh tiểu đường có giao diện thân thiện, tích hợp trong các phần mềm y tế để hỗ trợ bác sĩ trong chẩn đoán.

Tối ưu hóa thời gian xử lý để phù hợp với môi trường ứng dụng thời gian thực.

**Phân tích sâu hơn về các đặc trưng:**

Tăng cường các phương pháp phân tích tương quan và độ quan trọng của đặc trưng để xác định chính xác các yếu tố ảnh hưởng đến nguy cơ mắc bệnh.

Kết hợp với các nghiên cứu y học để bổ sung thông tin chuyên môn vào mô hình.