**BÁO CÁO CHI TIẾT**

**CHUYÊN NGÀNH CÔNG NGHỆ THÔNG TIN**

**XÂY DỰNG VÀ ỨNG DỤNG MÔ HÌNH HỌC MÁY DỰ ĐOÁN BỆNH TIỂU ĐƯỜNG**

|  |  |
| --- | --- |
| **SINH VIÊN THỰC HIỆN** | **CÁN BỘ HƯỚNG DẪN** |
| **LÊ ĐỨC NAM – 2221050299** | **TS.NGUYỄN HOÀNG LONG** |

MỤC LỤC

[1 GIỚI THIỆU ĐỀ TÀI 5](#_Toc186468185)

[2 MỤC TIÊU 6](#_Toc186468186)

[3 CÁC BƯỚC THỰC HIỆN 6](#_Toc186468187)

[3.1 LOAD DỮ LIỆU 6](#_Toc186468188)

[3.2 CHUẨN BỊ DỮ LIỆU 8](#_Toc186468189)

[3.2.1 Kiểm tra, xử lý trùng lặp 8](#_Toc186468190)

[3.2.2 Kiểm tra, xử lý khuyết thiếu 8](#_Toc186468191)

[3.2.3 Kiểm tra, xử lý lỗi cấu trúc 9](#_Toc186468192)

[3.3 LỰA CHỌN MÔ HÌNH 9](#_Toc186468193)

[3.3.1 Load các thư viện cần thiết 10](#_Toc186468194)

[3.3.2 Đặt biến mục tiêu và chia dữ liệu 10](#_Toc186468195)

[3.3.3 Xây dựng grid search 10](#_Toc186468196)

[3.3.4 Thực hiện feature selection 13](#_Toc186468197)

[3.4 TỐI ƯU MÔ HÌNH RANDOM FOREST 14](#_Toc186468198)

[3.4.1 Xây dựng mô hình random forest mới: 14](#_Toc186468199)

[3.4.2 Đánh giá mô hình mới: 15](#_Toc186468200)

[3.5 XÂY DỰNG ỨNG DỤNG 17](#_Toc186468201)

[4 TỔNG KẾT 18](#_Toc186468202)

[5 TÀI LIỆU THAM KHẢO 19](#_Toc186468203)

DANH MỤC HÌNH ẢNH

[Hình 1: Sức khỏe 5](#_Toc186468204)

[Hình 2: Những thư viện cần thiết 6](#_Toc186468205)

[Hình 3: Câu lệnh load và đọc dữ liệu 7](#_Toc186468206)

[Hình 4: Kết quả sau khi đọc dữ liệu 7](#_Toc186468207)

[Hình 5: Câu lệnh và kết quả kiểm tra trùng lặp 8](#_Toc186468208)

[Hình 6: Câu lệnh kiểm tra khuyết thiếu 8](#_Toc186468209)

[Hình 7: Câu lệnh kiểm tra kiểu dữ liệu 9](#_Toc186468210)

[Hình 8: Câu lệnh load các mô hình 10](#_Toc186468211)

[Hình 9: Đặt biến mục tiêu 10](#_Toc186468212)

[Hình 10: Chia dữ liệu 10](#_Toc186468213)

[Hình 11: Scoring 11](#_Toc186468214)

[Hình 12: Siêu tham số của từng mô hình 11](#_Toc186468215)

[Hình 13: Vòng lặp gridSearch 12](#_Toc186468216)

[Hình 14: Kết quả gridSearch 12](#_Toc186468217)

[Hình 15: Code feature importance 13](#_Toc186468218)

[Hình 16: feature importance 13](#_Toc186468219)

[Hình 17: Chia dữ liệu 14](#_Toc186468220)

[Hình 18: gridSearch cho mô hình random forest 14](#_Toc186468221)

[Hình 19: tham số tốt nhất cho mô hình random forest 14](#_Toc186468222)

[Hình 20: Câu lệnh tạo ma trận nhầm lẫn 15](#_Toc186468223)

[Hình 21: Ma trận nhầm lẫn 15](#_Toc186468224)

[Hình 22: Hàm tính toán các chỉ số đánh giá 16](#_Toc186468225)

[Hình 23: Bảng các chỉ số đánh giá 16](#_Toc186468226)

[Hình 24: Lưu mô hình 17](#_Toc186468227)

[Hình 25: Thư viện flask 17](#_Toc186468228)

[Hình 26: Gọi mô hình 17](#_Toc186468229)

[Hình 27: Tạo đường dẫn đến form nhập thông tin 18](#_Toc186468230)

[Hình 28: form người dùng nhập và kết quả từ mô hình 18](#_Toc186468231)

[Hình 29: Sử lý dự đoán 18](#_Toc186468232)

# GIỚI THIỆU ĐỀ TÀI

Bệnh tiểu đường là một trong những bệnh mãn tính phổ biến nhất trên toàn cầu, ảnh hưởng đến hàng triệu người mỗi năm và gây ra gánh nặng tài chính lớn đối với nền kinh tế. Để nghiên cứu và hiểu rõ hơn về các yếu tố nguy cơ liên quan đến bệnh tiểu đường, Trung tâm Kiểm soát và Phòng ngừa Dịch bệnh Hoa Kỳ (CDC) tiến hành Hệ thống Giám sát Yếu tố Nguy cơ Hành vi (BRFSS) – một khảo sát sức khỏe qua điện thoại được thực hiện hàng năm.



Hình 1: Sức khỏe

Trong dự án này, dữ liệu từ BRFSS năm 2015 được sử dụng, bao gồm một tệp CSV từ Kaggle. Tập dữ liệu này chứa phản hồi của 70962 cá nhân với 22 đặc điểm. Các đặc điểm này là câu trả lời từ những câu hỏi khảo sát với người tham gia, nhằm khám phá mối liên hệ giữa các yếu tố hành vi, sức khỏe và khả năng mắc bệnh tiểu đường.

Nguồn: <https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset>.

# MỤC TIÊU

Mục tiêu của dự án là khám phá và tìm hiểu dữ liệu nhằm hiểu rõ đặc điểm, cấu trúc cũng như các yếu tố ảnh hưởng đến việc dự đoán bệnh tiểu đường. Dựa trên sự phân tích dữ liệu, lựa chọn mô hình phù hợp để xây dựng, tối ưu hóa nhằm đảm bảo hiệu quả dự đoán và khả năng tổng quát hóa tốt nhất. Sau khi hoàn thiện mô hình, tiến hành đánh giá kết quả, đưa ra các nhận xét sâu sắc để cải thiện nếu cần. Cuối cùng, triển khai mô hình vào ứng dụng thực tế để giải quyết bài toán đặt ra một cách hiệu quả. Cụ thể:

* Khám phá, tìm hiểu dữ liệu.
* Chọn ra mô hình phù hợp với dữ liệu của vấn đề .
* Xây dựng, tối ưu mô hình.
* Đánh giá kết quả đưa ra nhận xét
* Đưa mô hình vào ứng dụng

# CÁC BƯỚC THỰC HIỆN

## LOAD DỮ LIỆU

Khi bắt đầu một dự án học máy, một trong những nhiệm vụ đầu tiên cần thực hiện chính là load dữ liệu. Đây là bước không thể thiếu và đóng vai trò cốt lõi trong toàn bộ quy trình phát triển mô hình. Dữ liệu là nguồn thông tin chủ yếu mà chúng ta sử dụng để huấn luyện mô hình, vì vậy việc load dữ liệu một cách hiệu quả sẽ giúp đảm bảo rằng mô hình được phát triển có thể đạt được độ chính xác cao nhất có thể.

A screen shot of a computer code

Description automatically generated

Hình 2: Những thư viện cần thiết

Giải thích ý nghĩa các câu lệnh:

* import pandas as pd : Import thư viện **Pandas** với biến là **pd**
* import matplotlib.pyplot as plt: mport module **pyplot** của **Matplotlib** với biến là **plt**
* import numpy as np: Import thư viện **NumPy** với biến là **np**
* pd.set\_option(‘display.max\_column’, None): Tăng giới hạn hiển thị số cột
* pd.set\_option(‘display.max\_colwidth’, None): Tăng giới hạn chiều rộng trong cột

A black screen with red text

Description automatically generated

Hình 3: Câu lệnh load và đọc dữ liệu

Sau khi thực hiện câu lệnh thì đã in ra màn hình thành công dữ liệu từ file CSV.

A screen shot of a black screen

Description automatically generated

Hình 4: Kết quả sau khi đọc dữ liệu

Bảng chứa dữ liệu liên quan đến các yếu tố hành vi, sức khỏe liên quan đến khả năng mắc bệnh tiểu đường. Dưới đây là các cột được mô tả trong bảng:

* Diabetes\_binary: Biến mục tiêu (1: Có bệnh tiểu đường, 0: Không có bệnh tiểu đường).
* HighBP: Huyết áp cao (1: Có, 0: Không).
* HighChol: Cholesterol cao (1: Có, 0: Không).
* CholCheck: Kiểm tra cholesterol gần đây (1: Có, 0: Không).
* BMI: Chỉ số khối cơ thể.
* Smoker: Tình trạng hút thuốc lá (1: Có, 0: Không).
* Stroke: Tiền sử đột quỵ (1: Có, 0: Không).
* HeartDiseaseorAttack: Tiền sử bệnh tim hoặc cơn đau tim (1: Có, 0: Không).
* PhysActivity: Thực hiện hoạt động thể chất thường xuyên (1: Có, 0: Không).
* Fruits: Tiêu thụ trái cây hàng ngày (1: Có, 0: Không).
* Veggies: Tiêu thụ rau củ hàng ngày (1: Có, 0: Không).
* HvyAlcoholConsump: Tiêu thụ rượu mạnh (1: Có, 0: Không).
* AnyHealthcare: Có tiếp cận dịch vụ y tế (1: Có, 0: Không).
* NoDocbcCost: Không thể gặp bác sĩ vì chi phí (1: Có, 0: Không).
* GenHlth: Tình trạng sức khỏe tổng quát (được đánh giá, có thể theo thang điểm từ 1-5).
* MentHlth: Sức khỏe tâm thần (số ngày không tốt về tâm lý trong tháng qua).
* PhysHlth: Sức khỏe thể chất (số ngày không tốt về thể chất trong tháng qua).
* DiffWalk: Khó khăn trong việc đi bộ (1: Có, 0: Không).
* Sex: Giới tính (0: Nam, 1: Nữ).
* Age: Tuổi .
* Education: Trình độ học vấn.
* Income: Nhóm thu nhập.

## CHUẨN BỊ DỮ LIỆU

### Kiểm tra, xử lý trùng lặp

Dữ liệu trùng lặp trong quá trình xử lý dữ liệu là các bản ghi giống nhau hoặc rất giống nhau xuất hiện nhiều lần trong tập dữ liệu. Sự trùng lặp này có thể xảy ra do nhiều nguyên nhân như lỗi khi nhập liệu, thu thập dữ liệu từ nhiều nguồn khác nhau, hoặc do việc sao chép dữ liệu. Việc tồn tại các bản ghi trùng lặp có thể gây ra nhiều vấn đề trong phân tích dữ liệu.

A screen shot of a computer

Description automatically generated

Hình 5: Câu lệnh và kết quả kiểm tra trùng lặp

Sau kiểm tra ta phát hiện trong tập dữ liệu có xuất hiện trùng lặp. Nhưng khi xét đến bài toán này, trùng lặp là tự nhiên và phản ánh tính thực tế, nên ta sẽ không loại bỏ chúng

### Kiểm tra, xử lý khuyết thiếu

A black screen with red and blue text

Description automatically generated

Hình 6: Câu lệnh kiểm tra khuyết thiếu

Việc kiểm tra dữ liệu khuyết thiếu là rất cần thiết vì có thể ảnh hưởng tới kết quả của quá trình phân tích dữ liệu. Sau khi kiểm tra, ta thấy trong dữ liệu không xuất hiện khuyết thiếu

### Kiểm tra, xử lý lỗi cấu trúc

A screenshot of a computer program

Description automatically generated

Hình 7: Câu lệnh kiểm tra kiểu dữ liệu

Sau khi sử dụng câu lệnh dtypes() ta thấy trong df đều thuộc dạng float và phù hợp với bài toán vậy nên không có lỗi kiểu dữ liệu.

## LỰA CHỌN MÔ HÌNH

Dựa trên việc phân tích dữ liệu, tập dữ liệu này chủ yếu bao gồm các đặc trưng nhị phân hoặc số học, và biến mục tiêu (Diabetes\_binary) thể hiện bài toán phân loại (Classification). Đây là bài toán thuộc nhóm học có giám sát (Supervised Learning), với mục tiêu dự đoán khả năng mắc bệnh tiểu đường dựa trên các yếu tố nguy cơ sức khỏe và hành vi. Để giải quyết bài toán, ta sẽ lựa chọn và thử nghiệm các mô hình phân loại quen thuộc đã học, bao gồm:

* Mô hình: Random Forest
* Mô hình: Logistic Regression
* Mô hình: Naive Bayes
* Mô hình: Decision Tree

Ở đây, để dễ dàng và thuận tiện cho việc lựa chọn mô hình, ta sẽ sử dụng kĩ thuật **grid search CV** để vừa có thể chọn ra mô hình phù hợp dựa vào chỉ số đánh giá, vừa có thể chọn ra những siêu tham số phù hợp cho mô hình.

### Load các thư viện cần thiết

Như đã liệt kê ở trên ta sẽ thực hiện load các mô hình để có thể thử nghiệm và chọn ra mô hình phù hợp nhất.

A screenshot of a computer program

Description automatically generated

Hình 8: Câu lệnh load các mô hình

* from sklearn.model\_selection import GridSearchCV, train\_test\_split: [GridSearchCV] Dùng để tìm kiếm siêu tham số tối ưu cho mô hình bằng kỹ thuật cross-validation. [train\_test\_split] hia dữ liệu thành tập huấn luyện và tập kiểm tra.
* from sklearn.ensemble import RandomForestClassifier: Nhập mô hình random forest
* from sklearn.linear\_model import LogisticRegression: Nhập mô hình Logistic regression
* from sklearn.naive\_bayes import GaussianNB: Nhập mô hình Naive Bayes
* from sklearn.tree import DecisionTreeClassifier: Nhập mô hình cây quyết định

### Đặt biến mục tiêu và chia dữ liệu

Ở bước này ta thực hiện lọc biến phụ thuộc và các biến độc lập ra thành x và y theo phân tích ở trên. y sẽ là biến mục tiêu (diabetes\_binary).

A black background with red and purple text

Description automatically generated

Hình 9: Đặt biến mục tiêu

Tiếp theo ta chia dữ liệu thành 2 mảng df\_X và df\_y ra làm 4 phần với tỉ lệ test là 25% cho dữ liệu test và 75% cho dữ liệu train với random state thống nhất là 42.



Hình 10: Chia dữ liệu

### Xây dựng grid search

Sau khi đã biết được những mô hình dùng để thử nghiệm với tập dữ liệu, ta thực hiện liệt kê các chỉ số đánh giá và các siêu tham số cho từng mô hình. Đối với các mô hình học máy phân loại có giám sát, ta sẽ sử dụng 4 chỉ số là accuracy, precision, recall và f1 để có thể so sánh từng mô hình.

A screen shot of a computer program

Description automatically generated

Hình 11: Scoring

Với mỗi mô hình ta sẽ sử dụng những tham số dánh riêng cho chúng và được giải thích ngay sau đây:

A computer screen shot of a program

Description automatically generated

Hình 12: Siêu tham số của từng mô hình

1. Mô hình Random forest:

* max\_depth: Độ sâu tối đa của mỗi cây
* min\_samples\_leaf: Số lượng mẫu tối thiểu tại một lá
* min\_samples\_split: Số lượng mẫu tối thiểu để tiếp tục phân tách tại một nút.
* max\_features: Số lượng đặc trưng (features) được sử dụng trong mỗi lần tách.
* n\_estimators: Số lượng cây trong rừng.

1. Mô hình LogisticRegression:

* C: Hệ số điều chỉnh phạt L2 (regularization)

1. Mô hình Naive Bayes:

* Mô hình này không cần tham số

1. Mô hình Decision Tree:

* max\_depth: Độ sâu tối đa của mỗi cây
* min\_samples\_split: Số lượng mẫu tối thiểu để tiếp tục phân tách tại một nút.

Sau khi có những tham số và chỉ số đánh giá cần thiết, ta thực hiện xây dựng vòng lặp chạy lần lượt với từng mô hình trong Dictionary models\_params. Trong vòng lặp này ta sẽ chia tập huấn luyện thành 5 fold (cv=5), sử dụng chỉ số f1 để tối ưu các mô hình rồi in hiệu xuất và các tham số của mô hình ra màn hình để so sánh.

A computer screen shot of text

Description automatically generated

Hình 13: Vòng lặp gridSearch

Sau khi chạy vòng lặp ta có kết quả sau:

A screen shot of a computer

Description automatically generated

Hình 14: Kết quả gridSearch

Dựa vào kết quả trên, ta có thể rút ra được là mô hình Random Forest là mô hình hiệu quả nhất cho tập dữ liệu

### Thực hiện feature selection

Chúng ta đã đưa ra kết luận là mô hình random forest là mô hình tốt nhất cho tập dữ liệu, vì vậy ta sẽ thực hiện feature selection dựa trên mô hình random forest rồi tiếp tục tối ưu mô hình này. Ở đây ta sẽ sử dụng phương pháp feature importance để có thể loại bỏ những cột không quan trọng trong mô hình nhằm giảm độ phức tạp của mô hình và tăng tính ứng dụng về sau.

A screen shot of a computer program

Description automatically generated

Hình 15: Code feature importance

Sau khi chạy code ta có kết quả:

A graph with blue bars

Description automatically generated

Hình 16: feature importance

Từ bảng trên ta có thể thấy Các feature GenHlth, HighBP, BMI, Age, và HighChol có Feature Importance cao nhất → Đây là các feature quan trọng cần được giữ lại. Còn các feature nằm ở dưới như AnyHealthcare, NoDocbCost, Fruits, Veggies, Smoker, Stroke, Sex, CholCheck. có Feature Importance gần như bằng 0. Điều này cho thấy chúng không đóng góp nhiều (hoặc hầu như không có đóng góp) vào khả năng phân loại của mô hình vậy nên ta sẽ loại bỏ chúng.

## TỐI ƯU MÔ HÌNH RANDOM FOREST

### Xây dựng mô hình random forest mới:

Sau khi ta đã xác định được những feature loại bỏ, ta sẽ thực hiện chia dữ liệu lại và sử dụng gridSearch để có thể tìm ra những tham số phù hợp cho mô hình này, và đây cũng là mô hình ta sẽ ứng dụng cho bên ngoài ở phần sau.



Hình 17: Chia dữ liệu

Sau khi đã có X, y mới ta thực hiện gridSearch cho mô hình:

A computer screen shot of a program code

Description automatically generated

Hình 18: gridSearch cho mô hình random forest

Sau khi chạy, ta đã có kết quả như sau: Mô hình có các tham số tốt nhất là max\_depth=5, max\_features=4, min\_samples\_leaf=3, n\_estimators=75.

A screenshot of a computer

Description automatically generated

Hình 19: tham số tốt nhất cho mô hình random forest

### Đánh giá mô hình mới:

Sau khi đã có mô hình mới ta sẽ xây dựng ma trận nhầm lẫn và 4 chỉ số đánh giá để đánh giá mô hình.

A computer screen shot of a code

Description automatically generated

Hình 20: Câu lệnh tạo ma trận nhầm lẫn

Sau khi chạy ta có ma trận:

A screenshot of a computer screen

Description automatically generated

Hình 21: Ma trận nhầm lẫn

Tiếp theo ta thực hiện tính toán các chỉ số đánh giá mô hình

A computer screen shot of a program code

Description automatically generated

Hình 22: Hàm tính toán các chỉ số đánh giá

Sau khi thực hiện hàm ta được bảng sau:

A screenshot of a computer

Description automatically generated

Hình 23: Bảng các chỉ số đánh giá

Từ ma trận nhầm lẫn và 4 chỉ số ta thấy rằng:

* True Negative (TN): 6117 Số lượng mẫu thuộc lớp 0 (âm tính) được dự đoán đúng
* False Positive (FP): 2720 Số lượng mẫu thuộc lớp 0 (âm tính) nhưng bị dự đoán sai thành lớp 1 (dương tính).
* False Negative (FN): 1875 Số lượng mẫu thuộc lớp 1 (dương tính) nhưng bị dự đoán sai thành lớp 0 (âm tính).
* True Positive (TP): 6961 Số lượng mẫu thuộc lớp 1 (dương tính) được dự đoán đúng
* Mô hình hoạt động khá tốt với accuracy ~73%, nhưng vẫn còn các trường hợp sai đáng kể (FP = 2720, FN = 1875).
* Precision (71%) cho thấy mô hình có tỷ lệ dự đoán dương tính đúng ở mức trung bình.
* Recall (78%) cao hơn, cho thấy mô hình bắt được phần lớn các mẫu dương tính.
* F1-Score (75%) cho thấy sự cân bằng giữa Precision và Recall, phù hợp cho bài toán cần quan tâm cả hai yếu tố.

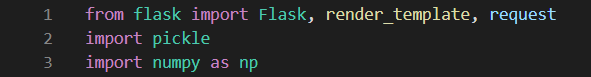
## XÂY DỰNG ỨNG DỤNG

Sau khi đã có mô hình thì ta sẽ chuyển mô hình thành ứng dụng để khách hàng có thể sử dụng hoặc áp dụng vào thực tế dự đoán. Trước hết ta sẽ sử dụng thư viện pickle của python để lưu mô hình để có thể dễ dàng sử dụng về sau.



Hình 24: Lưu mô hình

Tiếp đến ta sẽ sử dụng thư viện flask của python để xây dựng ứng dụng dự đoán bệnh tiểu đường bằng cách tạo 1 form cho người dùng nhập thông tin (lưu ý rằng những thông tin này phải phù hợp với những dữ liệu đầu vào của mô hình) sau đó dùng những dữ liệu đó chạy qua mô hình để có thể đưa ra dự đoán cho người dùng.



Hình 25: Thư viện flask

Ta sẽ sử dụng đường dẫn tuyệt đối để gọi ra mô hình đã lưu trước đó với biến là rf\_cv.



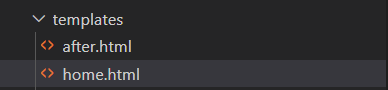
Hình 26: Gọi mô hình

Sau đó ta xây dựng form cho người dùng nhập thông tin để thu thập thông tin đó cho mô hình dự đoán và chỉnh đường dẫn flask vào thẳng form.

A black background with white text

Description automatically generated

Hình 27: Tạo đường dẫn đến form nhập thông tin



Hình 28: form người dùng nhập và kết quả từ mô hình

Người dùng nhập form và sau đó ta sẽ sử lý dự đoán qua form POST bằng mô hình.

Chúng ta có 13 biến đầu vào được chuyển đổi sang kiểu int và lưu vào 1 mảng rồi sử dụng mô hình rf\_cv để dự đoán. Kết quả sẽ được lưu trong pred.

A screen shot of a computer program

Description automatically generated

Hình 29: Sử lý dự đoán

Sau khi đã có kết quả ta sẽ trả về giao diện after.html với kết quả dự đoán (data=pred).

# TỔNG KẾT

* **Phân tích dữ liệu:** Tập trung xử lý dữ liệu sạch, kiểm tra các vấn đề như trùng lặp, khuyết thiếu và lỗi cấu trúc. Điều này đảm bảo chất lượng dữ liệu đầu vào cho các mô hình học máy.
* **Lựa chọn và tối ưu mô hình:** Random Forest được xác định là mô hình tốt nhất sau khi so sánh với Logistic Regression, Naive Bayes và Decision Tree. Quá trình tối ưu hóa các siêu tham số giúp cải thiện hiệu suất mô hình.
* **Hiệu quả mô hình:** Với các tham số tối ưu, mô hình đạt độ chính xác khoảng 73% và cân bằng giữa Precision (71%) và Recall (78%), phù hợp cho bài toán phân loại bệnh tiểu đường.
* **Triển khai thực tế:** Mô hình được triển khai thành ứng dụng dự đoán qua giao diện người dùng xây dựng bằng Flask, cho phép nhập thông tin và nhận kết quả dự đoán tức thì.

# TÀI LIỆU THAM KHẢO

* [Pawandeep-prog/deploy-ml-model-flask: deploy your ml model using flask](https://github.com/Pawandeep-prog/deploy-ml-model-flask)
* [py/ML/15\_gridsearch/exercise.md at master · codebasics/py](https://github.com/codebasics/py/blob/master/ML/15_gridsearch/exercise.md)