TRƯỜNG ĐẠI HỌC THỦY LỢI

KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN HỌC MÁY

**ĐỀ TÀI:**

**DỰ ĐOÁN NGUY CƠ TRẦM CẢM Ở SINH VIÊN**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:**  **Sinh viên:**  **Lớp:** | TS. Tạ Quang Chiểu  Nguyễn Thị Phương Anh  Nguyễn Minh Hiếu  64HTTT4 |

|  |  |
| --- | --- |
|  |  |

Hà Nội, 2024

LỜI NÓI ĐẦU

Sức khoẻ tinh thần đang ngày càng trở nên được quan tâm nhiều hơn, đặc biệt là về trầm cảm, bởi nó có thể gây ảnh hưởng nghiêm trọng tới chính chất lượng cuộc sống của người bệnh nói riêng và nền kinh thế, đời sống xã hội nói chung. Qua những số liệu thực tế, chúng em nhận thấy sự cần thiết trong việc ứng dụng thực tiễn và sự hữu ích đề tài này sẽ mang lại cho quá trình học tập của mình. Như vậy, chúng em đã quyết định triển khai đề tài *“Dự đoán nguy cơ trầm cảm ở sinh viên”* để dự đoán nguy cơ mắc bệnh trầm cảm ở sinh viên – một trong những đối tượng có nguy cơ trầm cảm cao nhất.

Mục tiêu chính của chúng em là áp dụng và kết hợp các thuật toán phân lớp để dự đoán các yếu tố quan trọng dẫn đến trầm cảm cũng như xây dựng mô hình có độ chính xác cao, tối ưu nhất.

Chúng em tin rằng đề tài sẽ giúp các trường học, cơ quan y tế, giáo dục có cái nhìn rõ ràng hơn về tình trạng trầm cảm ở sinh viên hiện nay, qua đó có thể phát hiện và can thiệp kịp thời.

Xin chân thành cảm ơn thầy Tạ Quang Chiểu đã tận tình hướng dẫn và hỗ trợ chúng em hoàn thành đề tài “*Dự đoán nguy cơ trầm cảm ở sinh viên*”.

Chúng em xin chân thành cảm ơn!

MỤC LỤC

[DANH MỤC CÁC HÌNH ẢNH v](#_Toc187184122)

[DANH MỤC BẢNG BIỂU vii](#_Toc187184123)

[CHƯƠNG 1 GIỚI THIỆU BÀI TOÁN 1](#_Toc187184124)

[1.1 Đặt vấn đề 1](#_Toc187184125)

[1.2 Phương pháp thực hiện 2](#_Toc187184126)

[CHƯƠNG 2 CƠ SỞ LÝ THUYẾT 3](#_Toc187184127)

[2.1 Cây quyết định (Decision Tree) 3](#_Toc187184128)

[2.2 Hồi quy logistic (Logistic regression) 4](#_Toc187184129)

[2.3 SVM (Support Vector Machine) 5](#_Toc187184130)

[2.4 Đánh giá kết quả mô hình 6](#_Toc187184131)

[2.4.1 Tính chính xác 6](#_Toc187184132)

[2.4.2 Ma trận nhầm lẫn 6](#_Toc187184133)

[2.4.3 Precision và recall 7](#_Toc187184134)

[2.4.4 K-fold cross-validation 7](#_Toc187184135)

[CHƯƠNG 3 DỮ LIỆU VÀ TIỀN XỬ LÝ DỮ LIỆU 8](#_Toc187184136)

[3.1 Nguồn dữ liệu 8](#_Toc187184137)

[3.2 Tiền xử lý dữ liệu 11](#_Toc187184138)

[3.2.1 Loại bỏ nhiễu và biến đổi dữ liệu 11](#_Toc187184139)

[3.2.2 Ma trận tương quan 20](#_Toc187184140)

[3.2.3 Dữ liệu đưa vào thực hiện 21](#_Toc187184141)

[CHƯƠNG 4 THỰC HIỆN VÀ ĐÁNH GIÁ KẾT QUẢ 22](#_Toc187184142)

[4.1 Sử dụng thuật toán Hồi quy tuyến tính và đánh giá kết quả 22](#_Toc187184143)

[4.1.1 Thực hiện 22](#_Toc187184144)

[4.1.2 Đánh giá kết quả 23](#_Toc187184145)

[4.2 So sánh với các mô hình khác 25](#_Toc187184146)

[4.2.1 So sánh độ chính xác giữa các mô hình 25](#_Toc187184147)

[4.2.2 Sử dụng k-fold crossvalidation 26](#_Toc187184148)

[CHƯƠNG 5 TRIỂN KHAI MÔ HÌNH 29](#_Toc187184149)

[TỔNG KẾT 32](#_Toc187184150)

[TÀI LIỆU THAM KHẢO 33](#_Toc187184151)

[PHỤ LỤC 34](#_Toc187184152)

DANH MỤC CÁC HÌNH ẢNH

Hình 3.1 Độ tin cậy của dữ liệu 8

Hình 3.2 Code đọc dữ liệu 8

Hình 3.3 Dữ liệu ban đầu 8

Hình 3.4 Code loại bỏ thuộc tính dư thừa 11

Hình 3.5 Code biến đổi dữ liệu 11

Hình 3.6 Kết quả sau khi xử lý 11

Hình 3.7 Code kiểm tra dữ liệu 11

Hình 3.8 Kết quả thu được 12

Hình 3.9 Code xử lý nhiễu 12

Hình 3.10 Kết quả thu được 13

Hình 3.11 Code kiểm tra dữ liệu 13

Hình 3.12 Kết quả thu được 13

Hình 3.13 Code kiểm tra và kết quả 14

Hình 3.14 Code thực hiện 14

Hình 3.15 Kiểm tra dữ liệu 14

Hình 3.16 Code thực hiện 14

Hình 3.17 Kết quả sau khi thực hiện 14

Hình 3.18 Kết quả thu được 15

Hình 3.19 Code thực hiện 15

Hình 3.20 Kết quả sau khi thực hiện 15

Hình 3.21 Số giá trị của thuộc tính 16

Hình 3.22 Xử lý nhiễu 16

Hình 3.23 Kết quả kiểm tra 16

Hình 3.24 Code xử lý 16

Hình 3.25 Code xử lý 16

Hình 3.26 Số lượng các giá trị trong thuộc tính 17

Hình 3.27 Code xử lý 17

Hình 3.28 Kết quả thu được 17

Hình 3.29 Code biến đổi dữ liệu 17

Hình 3.30 Kết quả 17

Hình 3.31 Dữ liệu ban đầu 17

Hình 3.32 Code xử lý 18

Hình 3.33 Kết quả thu được 18

Hình 3.34 Kết quả thu được 18

Hình 3.35 Code xử lý 18

Hình 3.36 Thuộc tính mới 18

Hình 3.37 Code biến đổi dữ liệu 18

Hình 3.38 Kết quả thu được 19

Hình 3.39 Code biến đổi dữ liệu 19

Hình 3.40 Kết quả thu được 19

Hình 3.41 Code biến đổi dữ liệu 19

Hình 3.42 Kết quả thu được 19

Hình 3.43 Code thực hiện 19

Hình 3.44 Code thực hiện 19

Hình 3.45 Dữ liệu sau khi xử lý 20

Hình 3.46 Code thực hiện 20

Hình 3.47 Kết quả thu được 20

Hình 3.48 Dữ liệu được đưa vào sử dụng 21

Hình 4.1 Code xác định biến mục tiêu 22

Hình 4.2 Code chia training set và test set 22

Hình 4.3 Code khởi tạo 22

Hình 4.4 Đưa vào dữ liệu huấn luyện 22

Hình 4.5 Đánh giá độ chính xác 22

Hình 4.6 Code biểu diễn ma trận nhầm lẫn 23

Hình 4.7 Ma trận nhầm lẫn 23

Hình 4.8 Code gọi mô hình 25

Hình 4.9 Tính toán và lưu trữ kết quả 25

Hình 4.10 Sắp xếp và biểu diễn dưới dạng biểu đồ 26

Hình 4.11 Biểu đồ thể hiện độ chính xác 26

Hình 4.12 Sử dụng thư viện 26

Hình 4.13 Khởi tạo k-fold 27

Hình 4.14 Code thực hiện 27

Hình 4.15 Code thực hiện 27

Hình 4.16 Kết quả thu được 28

Hình 5.1 Sử dụng thư viện 29

Hình 5.2 Xây dựng mô hình 29

Hình 5.3 Hàm chạy mô hình 29

Hình 5.4 Code nhập thông tin 30

Hình 5.5 Gọi hàm 30

Hình 5.6 Chạy thử mô hình 30

Hình 5.7 Chạy thử mô hình 30

Hình 5.8 Chạy thử mô hình 31

DANH MỤC BẢNG BIỂU

Bảng 2.1 Minh hoạ ma trận nhầm lẫn 6

Bảng 3.1 Bảng mô tả ý nghĩa các thuộc tính trong bộ dữ liệu 8

# GIỚI THIỆU BÀI TOÁN

## Đặt vấn đề

Trong những năm gần đây, cùng với sự phát triển của thời đại, tầm quan trọng của sức khoẻ tinh thần ngày càng nhận được nhiều sự quan tâm từ xã hội. Bất cứ ai, bất cứ độ tuổi hay ngành nghề nào cũng đều có khả năng trở thành nạn nhân của trầm cảm. Trầm cảm nếu không được phát hiện và có biện pháp can thiệp kịp thời sẽ dẫn đến những hậu quả ảnh hưởng tới cuộc sống của chính người bị trầm cảm như giảm năng lực học tập, làm việc, những trường hợp trầm cảm nặng còn dẫn tới xu hướng tự làm hại bản thân và thậm chí là tự sát [1]. Theo Tổ chức Y tế Thế giới (WHO), trầm cảm là nguyên nhân hàng đầu gây ra các vấn đề về sức khỏe tinh thần, ảnh hưởng đến hơn 280 triệu người trên toàn thế giới mỗi năm, trong đó sinh viên là nhóm đối tượng dễ bị tổn thương nhất [2]. Nghiên cứu từ Hiệp hội Tâm lý học Hoa Kỳ (APA) cho thấy hơn 41% sinh viên đại học tham gia báo cáo có các triệu chứng của lo âu hoặc trầm cảm [3].

Không chỉ là trên thế giới, tại Việt Nam, vấn đề sức khoẻ tinh thần ở sinh viên cũng là một vấn đề được nhiều người quan tâm. Năm 2021, Bộ Y tế và Bộ Giáo dục đã phối hợp thực hiện cuộc khảo sát trên 15.000 sinh viên từ các trường đại học cho thấy, khoảng 20-25% sinh viên có dấu hiệu của trầm cảm, lo âu hoặc căng thẳng kéo dài [4]. Những con số này minh chứng rõ ràng cho thực trạng đáng lo ngại của vấn đề sức khỏe tinh thần của sinh viên hiện nay.

Nhận thấy sự thiết thực và tính ứng dụng cho việc học tập của mình, nhóm chúng em đã quyết định triển khai đề tài *“Dự đoán nguy cơ trầm cảm ở sinh viên”* nhằm ứng dụng mô hình học máy vào quá trình dự đoán nguy cơ trầm cảm thay cho quá trình thực hiện khám bệnh thủ công tương đối khó thực hiện một cách phổ biến cho toàn bộ sinh viên trong một trường học nói riêng và phạm vi rộng hơn như cả nước nói chung. Việc tự động hóa quá trình dự đoán trầm cảm sẽ giúp cho những quy trình nghiên cứu, thống kê và khám chữa bệnh về sau được thực hiện một cách nhanh chóng hơn.

## Phương pháp thực hiện

1. Thu thập dữ liệu đầu vào
2. Tiền xử lý dữ liệu
3. Lựa chọn mô hình: Thử nghiệm với các thuật toán Cây quyết định (Decision Tree), SVM (Support Vector Machine), Hồi quy logistic (Logistic Regression)
4. Huấn luyện mô hình: Huấn luyện mô hình dựa trên tập huấn luyện
5. Đánh giá mô hình: Đánh giá độ hiệu quả và kết quả của mô hình
6. Triển khai mô hình

**Dữ liệu đầu vào:**

Bộ dữ liệu “Student Depression” bao gồm các đặc điểm như thông tin nhân khẩu học (tuổi, giới tính), kết quả học tập (điểm, chuyên cần), thói quen lối sống (thời gian ngủ, tập thể dục, hoạt động xã hội), tiền sử sức khỏe tâm thần và phản ứng với thang đo trầm cảm tiêu chuẩn [5]. Dữ liệu gồm 27900 hàng và 18 cột.

Nguồn dữ liệu: [Student Depression Dataset.](https://www.kaggle.com/datasets/hopesb/student-depression-dataset/data)

# CƠ SỞ LÝ THUYẾT

Bài toán dự đoán nguy cơ trầm cảm ở sinh viên có mục tiêu đầu ra là nhãn Depression Yes hoặc No, với đầu vào là các yếu tố liên quan trong dữ liệu ban đầu. Như vậy, ta áp dụng một số thuật toán phân loại nhị phân để giải quyết bài toán.

## Cây quyết định (Decision Tree)

ID3 (Iterative Dichotomiser 3) là một thuật toán học có giám sát (supervised learning) dùng để tạo ra cây quyết định. Nó hoạt động bằng cách chọn các thuộc tính tốt nhất để phân chia dữ liệu và xây dựng cây theo các giá trị của thuộc tính đó cho đến khi đạt được lá đại diện cho lớp của dữ liệu.

Để xét chọn các thuộc tính, ta sử dụng hàm số Entropy:

Cho một phân phối xác suất của một biến rời rạc ***x*** có thể nhận ***n*** giá trị khác nhau ***x1,x2,…,xn***. Giả sử rằng xác suất để ***x*** nhận các giá trị này là ***pi = p (x = xi)*** với ***0 ≤ pi ≤ 1,***  [6] . Ký hiệu phân phối này là ***p=(p1,p2,…,pn)***. Entropy của phân phối này được định nghĩa là

*(2-1)*

trong đó log là logarit tự nhiên.

Xét bài toán với *C*class khác nhau, giả sử ta đang làm việc với một *non-leaf* với các điểm dữ liệu tạo thành một tập *S* với số phần tử là ***|S| = N****.* Giả sử thêm rằng trong số N điểm dữ liệu này, ***Nc , c = 1, 2, …, C*** điểm thuộc vào class c. Xác suất để mỗi điểm dữ liệu rơi vào một class s được xấp xỉ bằng (maxium likehood estimation) [6]

Entropy tại

*(2-2)*

Giả sử thuộc tính được chọn là ***x.*** Dựa trên ***x,*** các điểm dữ liệu trong *S* được phân ra thành *K* child note S1, S2, …, Sk với số điểm trong mỗi child node lần lượt là m1, m2, …, mk.. Ta định nghĩa

*(2-3)*

Ta định nghĩa *information gain* dựa trên thuộc tính x

***G(x, S) = H(S) – H(x, S)*** *(2-4)*

Tại mỗi node, thuộc tính được chọn được xác định dựa trên

***x\* = argmax G(x,S) = argmin H(x,S)****(2-5)*

Các bước hoạt động của thuật toán [7]:

Algorithm GenDecTree(Sample S, Attlist A)

1. Tạo một nút N
2. Nếu tất cả các mẫu thuộc cùng lớp C thì N được gán nhãn C; dừng thuật toán
3. Nếu A là rỗng thì N được gán nhãn C là nhãn phổ biến nhất trong S; dừng thuật toán;
4. Chọn a ∈ A có độ đo *information gain* cao nhất; gán nhãn N theo a;
5. Với mỗi giá trị v của a:
   1. Phát triển 1 nhánh từ N với điều kiện a = v
   2. Đặt Sv là tập con của S với a = v
   3. Nếu Sv là rỗng thì gắn một lá có nhãn phổ biến nhất trong S
   4. Ngược lại gắn một nút được tạo bởi GenDecTree(Sv, A – a)

## Hồi quy logistic (Logistic regression)

Mặc dù có tên gọi là hồi quy nhưng thuật toán này lại được sử dụng trong các bài toán phân lớp không phải giá trị liên tục do nó có đầu ra dưới dạng nhị phân 0 hoặc 1. Hồi quy logistic có dạng *y* *= f (****wT x),*** trong đó *f (****wT x***) là hàm kích hoạt (activation function).

Để giới hạn kết quả đầu ra, ta sử dụng hàm sigmoid:

*(2-6)*

Hàm này bị chặn trong khoảng (0, 1), phù hợp với yêu cầu đầu ra của thuật toán.

Hàm mất mát:

P(yi | xi; w) = zyii (1 – zi)1-yi *(2-7)*

Ta cầm tìm w để hàm số P(y | X; w) trên toàn bộ training set đạt giá trị lớn nhất

J(w) = -log P(y|X;w)

= *(2-8)*

hay

w = w + η(yi − zi)xi *(2-9)*

Sau khi có w, ta dự đoán nhãn

ŷ = *sigmoid*(w0 + w1x1 + w2x2 + … + wdxd) *(2-10)*

Nếu ŷ > 0.5 thì x thuộc về lớp 1, ngược lại, x thuộc về lớp 0.

## SVM (Support Vector Machine)

SVM tìm kiếm một siêu phẳng (hyperplane) để phân tách dữ liệu thuộc hai lớp khác nhau sao cho khoảng cách từ siêu phẳng này đến các điểm dữ liệu gần nhất của mỗi lớp là lớn nhất.

Một siêu phẳng trong không gian 𝑛 chiều có thể được biểu diễn dưới dạng

***w \* x + b = 0*** *(2-11)*

trong đó

* ***w:*** vector trọng số
* ***x:*** vector dữ liệu
* ***b:***bias

Khoảng cách từ một điểm đến siêu phẳng được tính bằng công thức

*(2-12)*

Trong đó w = [w0, w1, …, wd]T

Nếu bỏ dấu trị tuyệt đối ở tử số, ta biết được điểm đó nằm về phía nào của mặt phẳng đang xét:

* Những điểm mang dấu dương nằm về cùng 1 phía
* Những điểm mang dấu âm nằm về phía còn lại
* Những điểm nằm trên mặt phẳng làm cho tử số có giá trị bằng 0, tức khoảng cách bằng 0

Bài toán tối ưu:

*(2-13)*

thoả mãn: 1 – yn (wTxn + b) 0, ∀ n = 1, 2, …, N

Sau khi đã tìm được mặt phân cách wTx + b = 0

Nhãn của bất kỳ một diểm được xác định bằng class(x) = sgn(wTx + b)

## Đánh giá kết quả mô hình

### Tính chính xác

### Ma trận nhầm lẫn

* TPi (true positive): Số lượng dữ liệu thuộc lớp ci được phân loại chính xác vào lớp ci
* FPi (false positive): Số lượng dữ liệu bên ngoài bị phân loại nhầm vào lớp ci
* TNi (true negative): Số lượng dữ liệu không thuộc lớp ci được phân loại (chính xác)
* FNi (false negative): Số lượng dữ liệu thuộc lớp ci bị phân loại nhầm (vào các lớp khác ci)

Bảng . Minh hoạ ma trận nhầm lẫn

|  |  |  |  |
| --- | --- | --- | --- |
| Lớp ci | | Được phân loại bởi hệ thống | |
| Thuộc | Không thuộc |
| Nhãn lớp đúng | Thuộc | TPi | FNi |
| Không thuộc | FPi | TNi |

### Precision và recall

Precision đối với lớp ci: Tổng số các ví dụ thuộc lớp ci phân loại chính xác được chia cho tổng số các ví dụ được phân loại vào lớp ci

Precision(ci) = TPi / (TPi +FPi)

Recall đối với lớp ci: Tổng số các ví dụ thuộc lớpci phân loại chính xác được chia cho tổng số các ví dụ thuộc lớp ci

Recall(ci) = TPi / (TPi +FNi)

Tiêu chí đánh giá F1 là sự kết hợp của 2 tiêu chí đánh giá Precision và Recall

F1 = 2\*(Precision \* Recall)/(Precision + Recall)

### K-fold cross-validation

K-fold cross-validation là một phương pháp đánh giá mô hình phổ biến, chia dữ liệu thành 𝐾 tập con (folds) và sử dụng 𝐾 – 1 tập để huấn luyện và 1 tập để kiểm tra, lặp lại 𝐾 lần. Điều này giúp đánh giá hiệu suất mô hình ổn định hơn so với việc chỉ chia dữ liệu thành một tập huấn luyện và kiểm tra duy nhất.

# DỮ LIỆU VÀ TIỀN XỬ LÝ DỮ LIỆU

## Nguồn dữ liệu

Bộ dữ liệu “Student Depression” được lấy trên Kaggle với 27900 hàng và 18 cột, được nhiều người tin cậy và sử dụng.

A screenshot of a graph

Description automatically generated

Hình . Độ tin cậy của dữ liệu

Nguồn dữ liệu: [Student Depression Dataset.](https://www.kaggle.com/datasets/hopesb/student-depression-dataset/data)



Hình . Code đọc dữ liệu

A table with many small text

Description automatically generated with medium confidence

Hình . Dữ liệu ban đầu

**Giải thích thuộc tính**

Bảng . Bảng mô tả ý nghĩa các thuộc tính trong bộ dữ liệu

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Mô tả** | **Ý nghĩa** |
| 1 | id | Định danh cho từng sinh viên thực hiện khảo sát | Được sử dụng để phân biệt từng cá nhân trong dataset |
| 2 | age | Độ tuổi của người thực hiện khảo sát, thể hiện dưới dạng số nguyên | Một số độ tuổi nhất định có nguy cơ cao hơn đối với các vấn đề tâm lý |
| 3 | gender | Giới tính (Male/Female) | Giới tính có thể ảnh hưởng đến mức độ nhạy cảm với trầm cảm |
| 4 | city | Thành phố nơi cá nhân đang sinh sống | Yếu tố địa lý có thể ảnh hưởng đến môi trường sống và áp lực |
| 5 | CGPA | Điểm trung bình tích lũy của sinh viên, ở đây được đánh giá trên thang điểm từ 0 đến 10 | Hiệu suất học tập có thể liên quan đến sự căng thẳng của cá nhân |
| 6 | Sleep duration | Số giờ ngủ trung bình mỗi ngày | Giấc ngủ là yếu tố quan trọng ảnh hưởng đến sức khỏe tinh thần |
| 7 | Profession | Nghề nghiệp hiện tại của cá nhân | Nghề nghiệp liên quan đến mức độ áp lực và sức khỏe tinh thần |
| 8 | Work pressure | Mức độ áp lực trong công việc, được đánh giá trên thang từ 0 đến 5 | Ảnh hưởng trực tiếp đến sức khỏe tinh thần của người lao động |
| 9 | Academic pressure | Mức độ áp lực học tập (được đánh giá trên thang từ 0 đến 5) | Là một yếu tố chính gây căng thẳng và trầm cảm, đặc biệt ở sinh viên |
| 10 | Study Satisfaction | Mức độ hài lòng với việc học (được đánh giá trên thang từ 0 đến 5) | Hài lòng với việchọc có thể giảm nguy cơ trầm cảm |
| 11 | Job Satisfaction | Mức độ hài lòng với công việc (được đánh giá trên thang từ 0 đến 5) | Sự hài lòng trong công việc ảnh hưởng lớn đến sức khỏe tinh thần |
| 12 | Dietary habit | Chế độ ăn uống của cá nhân (lành mạnh hoặc không lành mạnh). | Thói quen ăn uống ảnh hưởng đến sức khỏe cả thể chất và tinh thần |
| 13 | Degree | Trình độ học vấn hiện tại | Mức học vấn có thể liên quan đến áp lực học tập |
| 14 | Have you ever had suicidal thoughts? | Câu trả lời Có/Không về việc từng có ý định tự sát. | Là dấu hiệu cảnh báo quan trọng cho các vấn đề sức khỏe tinh thần nghiêm trọng |
| 15 | Work/Study Hours | Số giờ làm việc hoặc học tập mỗi ngày. | Làm việc hoặc học tập quá mức có thể dẫn đến căng thẳng và kiệt sức |
| 16 | Financial Stress | Mức độ căng thẳng do vấn đề tài chính (được đánh giá trên thang từ 0 đến 5) | Các vấn đề tài chính có thể là nguồn áp lực lớn |
| 17 | Family History of Mental Illness | Có hoặc không có tiền sử gia đình mắc bệnh tâm thần | Là một yếu tố nguy cơ tiềm ẩn đối với trầm cảm |
| 18 | Depression | Phân loại Yes hoặc No | Là biến mục tiêu, được sử dụng để phân tích và dự đoán |

## Tiền xử lý dữ liệu

Với bước tiền xử lý dữ liệu, ta thực hiện các công việc loại bỏ những giá trị nhiễu, dư thừa và biến đổi dữ liệu cho từng thuộc tính.

### Loại bỏ nhiễu và biến đổi dữ liệu

Xét thuộc tính id, ta thực hiện loại bỏ thuộc tính này vì nó không ảnh hưởng tới kết quả đầu ra



Hình . Code loại bỏ thuộc tính dư thừa

Thực hiện biến đổi dữ liệu của thuộc tính “*Gender*” để giúp cho việc xử lý dữ liệu dễ dàng hơn. Ta biến đổi giá trị “Male” thành 0 và “Female” thành 1



Hình . Code biến đổi dữ liệu

Kết quả sau xử lý

A screenshot of a computer

Description automatically generated

Hình . Kết quả sau khi xử lý

Đối với thuộc tính “*City*”, ta kiểm tra và đếm số giá trị trong dữ liệu



Hình . Code kiểm tra dữ liệu

Kết quả thu được

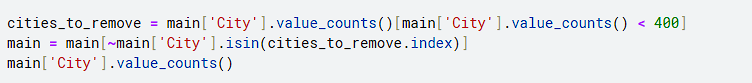
A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Hình . Kết quả thu được

Để tăng độ phổ biến và chính xác cho mô hình, ta loại bỏ những thành phố có số lượng thu được dưới 400



Hình . Code xử lý nhiễu

Kết quả thu được

A screenshot of a computer

Description automatically generated

Hình . Kết quả thu được

Đối với thuộc tính “*Profession”,* ta thực hiện kiểm tra dữ liệu ban đầu



Hình . Code kiểm tra dữ liệu

Kết quả thu được

A screenshot of a computer

Description automatically generated

Hình . Kết quả thu được

Có thể thấy rằng dữ liệu chứa cả những ngành nghề không phải sinh viên (đối tượng chính đề tài muốn hướng dến), ta lọc những dữ liệu mang giá trị “Student”:

*A computer screen shot of a computer code

Description automatically generated*

Hình 3.13 Code kiểm tra và kết quả

Tuy nhiên, nếu một cột dữ liệu chỉ mang một giá trị thì nó không có ý nghĩa trong việc xây dựng mô hình, ta thực hiện loại bỏ thuộc tính “*Profession”*

**

Hình 3.14 Code thực hiện

Ta tiếp tục xét đến thuộc tính “*Work pressure*”, vì đã lọc bỏ những ngành nghề khác ngoài sinh viên nên giá trị của thuộc tính này gần như 100% là 0, ta thực hiện loại bỏ thuộc tính vì nó không cần thiết

A screenshot of a computer

Description automatically generated

Hình . Kiểm tra dữ liệu

Thực hiện loại bỏ thuộc tính “*Work pressure”*

**

Hình 3.16 Code thực hiện

A screenshot of a computer

Description automatically generated

Hình . Kết quả sau khi thực hiện

Thực hiện kiểm tra dữ liệu với thuộc tính “*Age”,* ta thu được kết quả:

A screenshot of a number

Description automatically generatedA screenshot of a number

Description automatically generated

Hình . Kết quả thu được

Loại bỏ những dữ liệu có độ tuổi lớn hơn 30 do những giá trị còn lại có số lượng dữ liệu rất ít và có thể gây nhiễu cho quá trình chạy mô hình



Hình . Code thực hiện

A number in a row

Description automatically generated with medium confidence

Hình . Kết quả sau khi thực hiện

Kiểm tra thuộc tính “*Academic pressure”,* ta thu được

A screenshot of a computer

Description automatically generated

Hình . Số giá trị của thuộc tính

Thực hiện lược bỏ dữ liệu mang giá trị 0 (vì số lượng của giá trị này quá ít) bằng cách giữ lại những giá trị lớn hơn 0



Hình . Xử lý nhiễu

Tiếp tục kiểm tra thuộc tính “Study Satisfaction”

A number and numbers on a white background

Description automatically generated

Hình . Kết quả kiểm tra

Loại bỏ giá trị nhiễu



Hình . Code xử lý

Thực hiện loại bỏ cột “*Job Satisfaction”* do đã loại bỏ cột “*Work pressure”*



Hình . Code xử lý

Thực hiện kiểm tra dữ liệu “*Sleep Duration”*, nhận thấy rằng giá trị “Other” có thể gây nhiễu, ta thực hiện lược bỏ chúng

A white background with black text

Description automatically generated

Hình . Số lượng các giá trị trong thuộc tính

Thực hiện và kiểm tra kết quả



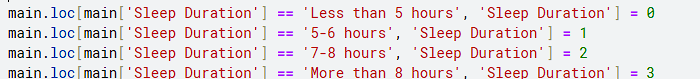
Hình . Code xử lý

A white background with black text

Description automatically generated

Hình . Kết quả thu được

Thực hiện biến đổi các giá trị trên thành giá trị dạng số để đơn giản hoá dữ liệu



Hình . Code biến đổi dữ liệu

A number of numbers and words

Description automatically generated with medium confidence

Hình . Kết quả

Xét thuộc tính “*Dietary Habits*”, ta loại bỏ thuộc tính gây nhiễu và thực hiện biến đổi dữ liệu sang dạng số

A white background with black text

Description automatically generated

Hình . Dữ liệu ban đầu

A screenshot of a computer

Description automatically generated

Hình . Code xử lý

A black text on a white background

Description automatically generated

Hình . Kết quả thu được

Kiểm tra giá trị của thuộc tính “*Degree*”



ta thu được

A group of black text

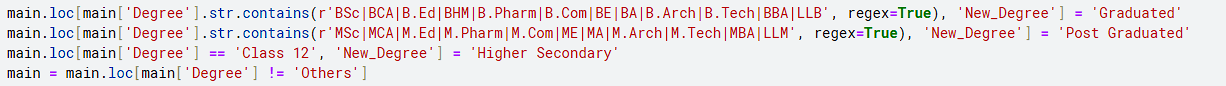
Description automatically generated

Hình . Kết quả thu được

Để thuận tiện cho việc xử lý dữ liệu về sau, ta tạo một thuộc tính mới “*New\_Degree*”, dữ liệu trong thuộc tính mới sẽ được tổng hợp từ thuộc tính “*Degree*”.

Giá trị của mới “*New\_Degree*” gồm có “Graduated”, “Post Graduated” và “Higher Secondary”.

Đồng thời, ta loại bỏ giá trị “Other” của cột “*Degree*”.



Hình . Code xử lý

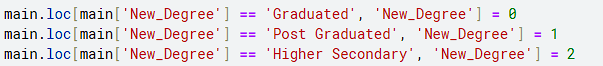
Thuộc tính mới:

A white background with black text

Description automatically generated

Hình . Thuộc tính mới

Thực hiện biến đổi giá trị trên thành dạng số để dễ dàng xử lý



Hình . Code biến đổi dữ liệu

A number on a white background

Description automatically generated

Hình . Kết quả thu được

Xét thuộc tính “*Have you ever had suicidal thought?*”, ta biến đổi giá trị trong thuộc tính này thành dạng số



Hình . Code biến đổi dữ liệu

A close-up of a white background

Description automatically generated

Hình . Kết quả thu được

Xét thuộc tính *“Work/Study Hours”* và *“Financial Stress”* không có giá trị nào gây nhiễu.

Với thuộc tính “”, ta biến đổi dữ liệu thành dạng số



Hình . Code biến đổi dữ liệu

A close-up of a text

Description automatically generated

Hình . Kết quả thu được

Sau khi đã xem xét tất cả các thuộc tính, ta kiểm tra xem trong dữ liệu có dữ liệu nào bị thiếu (mang giá trị null) hay không và loại bỏ chúng

****

Hình . Code thực hiện

Thực hiện xoá bỏ cột “*City*” và “*Degree*” không còn cần thiết

A close-up of a computer code

Description automatically generated

Hình . Code thực hiện

Dữ liệu thu được

A table with numbers and text

Description automatically generated with medium confidence

Hình . Dữ liệu sau khi xử lý

### Ma trận tương quan

A close up of a computer screen

Description automatically generated

Hình . Code thực hiện

Ta sử dụng ma trận tương quan dưới dạng heatmap nhằm xác định mối quan hệ giữa các thuộc tính trong dữ liệu

A graph with red and blue squares

Description automatically generated

Hình . Kết quả thu được

Ở biểu đồ này, mối quan hệ giữa các thuộc tính có màu càng nóng (hướng về màu đỏ) thì các thuộc tính đó có xu hướng biến đổi cùng chiều với nhau, ngược lại, chúng sẽ biến đổi ngược chiều với màu càng lạnh (màu xanh).

Như vậy, ta có thể thấy các thuộc tính “*Academic pressure”, “Have you ever had suicidal thought?”* và *“Finacial stress”* có mức độ tương quan với thuộc tính mục tiêu “*Depression*” hơn các thuộc tính khác.

### Dữ liệu đưa vào thực hiện

Dữ liệu sau khi tiền xử lý gồm 20501 hàng và 13 cột, gồm có các thuộc tính: *'Gender', 'Age'', 'Academic Pressure', 'CGPA', 'Study Satisfaction', 'Sleep Duration', 'Dietary Habits', 'Degree', 'Have you ever had suicidal thoughts ?', 'Work/Study Hours', 'Financial Stress', 'Family History of Mental Illness', 'Depression', 'New\_Degree'*

A table with numbers and text

Description automatically generated with medium confidence

Hình . Dữ liệu được đưa vào sử dụng

# THỰC HIỆN VÀ ĐÁNH GIÁ KẾT QUẢ

## Sử dụng thuật toán Hồi quy tuyến tính và đánh giá kết quả

### Thực hiện

Đầu tiên, ta loại bỏ giá trị trong thuộc tính “*Depression*” do đây là thuộc tính mục tiêu

Hình . Code xác định biến mục tiêu

Chia dữ liệu thành tập huấn luyện và tập kiểm tra, tập kiểm tra sẽ chiếm 20% dữ liệu đưa vào ban đầu



Hình . Code chia training set và test set

A close-up of a computer code

Description automatically generated

Hình . Code khởi tạo

StandardScaler(): Tạo một đối tượng StandardScaler. Phương pháp này chuẩn hóa dữ liệu sao cho mỗi đặc trưng có trung bình bằng 0 và độ lệch chuẩn bằng 1.

scaler.fit\_transform(X\_train): Tính toán các tham số chuẩn hóa (trung bình và độ lệch chuẩn) từ tập huấn luyện X\_train và áp dụng chuẩn hóa cho dữ liệu huấn luyện.

scaler.transform(X\_test): Áp dụng cùng các tham số chuẩn hóa (được tính từ X\_train) lên dữ liệu kiểm tra X\_test.

Ta sử dụng thuật toán Hồi quy tuyến tính với dữ liệu huấn luyện

**A close up of a word

Description automatically generated**

Hình . Đưa vào dữ liệu huấn luyện

Đưa ra độ chính xác





Hình . Đánh giá độ chính xác

### Đánh giá kết quả

Ta có thể thấy độ chính xác khá cao khi ứng dụng Hồi quy tuyến tính với dữ liệu này đạt 85.29%.

Tiếp theo, ta xét ma trận nhầm lẫn (confusion matrix)

A close-up of a computer screen

Description automatically generated

Hình 4.6 Code biểu diễn ma trận nhầm lẫn

Kết quả thu được

A blue squares with white text

Description automatically generated

Hình 4.7 Ma trận nhầm lẫn

Dựa vào ma trận nhầm lẫn trong hình, ta có thể thấy rằng:

* True Positives (TP): Số trường hợp dự đoán đúng là lớp 1 (thực tế là 1 và dự đoán là 1). Trong ma trận, đây là ô chứa giá trị 2389.
* True Negatives (TN): Số trường hợp dự đoán đúng là lớp 0 (thực tế là 0 và dự đoán là 0). Trong ma trận, đây là ô chứa giá trị 1106.
* False Positives (FP): Số trường hợp dự đoán là lớp 1 nhưng thực tế là lớp 0 (thực tế là 0 nhưng dự đoán là 1). Trong ma trận, đây là ô chứa giá trị 365.
* False Negatives (FN): Số trường hợp dự đoán là lớp 0 nhưng thực tế là lớp 1 (thực tế là 1 nhưng dự đoán là 0). Trong ma trận, đây là ô chứa giá trị 238.

Thực hiện tính toán các độ đo đánh giá, ta có:

**Accuracy** (Độ chính xác) = (TP + TN)/(TP + TN + FP + FN)

= (2389 + 1106)/(2389 + 1106 + 365 + 238)

= 0.8529 (85.29%)

Đây là tỷ lệ phần trăm các dự đoán đúng (cả True Positives và True Negatives) so với tổng số dự đoán. Độ chính xác khá cao, cho thấy mô hình có khả năng phân loại tốt hầu hết các trường hợp.

**Precision** (Độ chính xác) = TP/(TP + FP)

= 2389/(2389 + 365)

= 0.8675 (86.75%)

Độ chính xác cho lớp 1 cho biết tỷ lệ phần trăm các dự đoán lớp 1 là đúng. Với giá trị cao như vậy, mô hình ít có khả năng đưa ra dự đoán sai về lớp 1 (ít False Positives).

**Recall** (Độ hồi tưởng) = TP/(TP + FN)

= 2389/(2389 + 238)

= 0.9094 (90.94%)

Độ hồi tưởng cho lớp 1 cho biết tỷ lệ phần trăm các trường hợp thực tế là lớp 1 mà mô hình dự đoán đúng. Giá trị cao của recall cho thấy mô hình nhận diện tốt các trường hợp thực sự là lớp 1, nhưng vẫn còn một số ít bị bỏ sót (False Negatives).

**F1-Score** = 2 \* (Precision \* Recall)/(Precision + Recall)

= 2 \* (0.8675 \* 0.9094)/(0.8675 + 0.9094)

= 0.8879 (88.79%)

Đây là chỉ số kết hợp giữa precision và recall, thể hiện sự cân bằng giữa hai chỉ số này. Với F1-Score cao, mô hình đạt hiệu quả tốt trong việc cân bằng giữa độ chính xác và độ hồi tưởng.

Mô hình này hoạt động khá tốt, với các chỉ số đều trên 85%. Điều này cho thấy mô hình không chỉ chính xác mà còn nhất quán trong việc phân loại hai lớp.

Tuy nhiên, có một số trường hợp mà mô hình dự đoán sai (365 False Positives và 238 False Negatives), nhưng so với tổng số dữ liệu, tỷ lệ này khá thấp.

## So sánh với các mô hình khác

Để cải thiện độ chính xác cho bài toán, ta thực hiện đánh giá với các thuật toán Cây quyết định (Decision Tree) và SVM (Support Vector Machine) bằng cách sử dụng thư viện sklearn trong Python

A screenshot of a computer program

Description automatically generated

Hình 4.8 Code gọi mô hình

### So sánh độ chính xác giữa các mô hình

Tính toán độ chính xác và lưu trữ kết quả

A screen shot of a computer code

Description automatically generated

Hình . Tính toán và lưu trữ kết quả

Sắp xếp kết độ chính xác theo thứ tự giảm dần và biểu diễn dưới dạng biểu đồ

A computer screen shot of a code

Description automatically generated

Hình . Sắp xếp và biểu diễn dưới dạng biểu đồ

Kết quả thu được

A graph showing different colored squares

Description automatically generated with medium confidence

Hình . Biểu đồ thể hiện độ chính xác

Như vậy, ta có thể thấy SVM đang cho kết quả có độ chính xác cao nhất, Hồi quy logistic cho độ chính xác tương đương và Cây quyết định có độ chính xác thấp nhất.

### Sử dụng k-fold crossvalidation

Sử dụng thư viện sklearn để tính toán và matplotib, seaborn để vẽ biểu đồ

A screenshot of a computer

Description automatically generated

Hình . Sử dụng thư viện

Khởi tạo k-fold với k=5



Hình . Khởi tạo k-fold

Lưu kết quả accuracy và thực hiện vòng lặp đánh giá tập huấn luyện (training set)

A screen shot of a computer code

Description automatically generated

Hình . Code thực hiện

Biểu diễn bằng biểu đồ

A computer screen shot of a code

Description automatically generated

Hình . Code thực hiện

Kết quả thu được

A graph of a graph with numbers and lines

Description automatically generated with medium confidence

Hình . Kết quả thu được

Ta có thể thấy được hiệu suất giữa các mô hình:

* Hồi quy logistic (Logistic Regression): Có độ chính xác ổn định và cao nhất trong tất cả các tỷ lệ dữ liệu. Điều này cho thấy bài toán có thể được giải quyết hiệu quả bằng mô hình tuyến tính và hồi quy logistic là một lựa chọn phù hợp.
* Cây quyết định (Decision Tree): Độ chính xác ở mức trung bình, thấp hơn Logistic Regression. Mô hình Cây quyết định (Decision Tree) có thể đã bị overfitting trên các tập dữ liệu nhỏ (với tỷ lệ kích thước tập huấn luyện thấp) do đặc tính dễ phân nhánh quá mức.
* SVM: Hiệu suất tốt trên các tập dữ liệu nhỏ nhưng có thể không cao bằng Logistic Regression trên tập dữ liệu lớn. Điều này có thể do việc lựa chọn kernel chưa tối ưu hoặc SVM cần tinh chỉnh thêm siêu tham số để phù hợp hơn.

Dựa vào biểu đồ có thể thấy hồi quy logistic có xu hướng tăng đều và đạt hiệu suất cao nhất, Cây quyết định và SVM có sự dao động ở một vài mức dữ liệu nhỏ, nhưng dần ổn định khi dữ liệu lớn hơn.

Như vậy, ta sử dụng thuật toán hồi quy logistic (Logistic regression) xây dựng mô hình để giải quyết bài toán ban đầu.

# TRIỂN KHAI MÔ HÌNH

Ta triển khai mô hình với thông tin đầu vào được nhập bởi người dùng

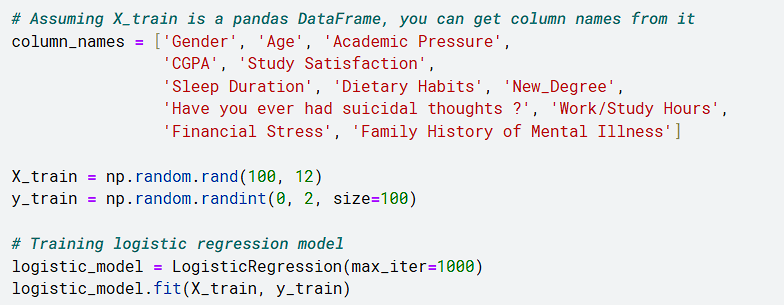
Sử dụng các thư viện pandas, numpy và sklearn để tính toán

A screen shot of a computer

Description automatically generated

Hình . Sử dụng thư viện

Xây dựng mô hình



Hình . Xây dựng mô hình

Hàm trả về kết quả dự đoán

A screenshot of a computer program

Description automatically generated

Hình . Hàm chạy mô hình

A close-up of a text

Description automatically generated

Hình . Code nhập thông tin

Gọi hàm trả về kết quả



Hình . Gọi hàm

Kiểm tra mô hình:

A white text with black text

Description automatically generated

Hình . Chạy thử mô hình

A computer code with numbers and letters

Description automatically generated

Hình . Chạy thử mô hình

A computer code with black text

Description automatically generated

Hình . Chạy thử mô hình

TỔNG KẾT

Sau khi hoàn thành đề tài này, chúng em đã nắm bắt được những kiến thức quan trọng về các thuật toán trong phạm vi môn học học máy, đặc biệt là các thuật toán hồi quy logistic (Logistic regression), cây quyết định (Decision tree), SVM (support vector machine) cũng như cách đánh giá độ chính xác của mô hình.

Ngoài ra, chúng em cũng học được cách làm việc nhóm hiệu quả, phân chia công việc hợp lý, và phối hợp để đạt được mục tiêu chung. Những kinh nghiệm từ đề tài này không chỉ giúp chúng em nắm bắt lý thuyết mà còn mang lại kỹ năng quan trọng để áp dụng cho những bài toán thực tế cũng như những đề tài, công việc trong tương lai.

TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Tổ chức Y tế Thế giới (WHO), "Depression," 2012. |
| [2] | Tổ chức Y tế Thế giới (WHO), "Depression," 2023. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/depression. |
| [3] | Hiệp hội Tâm lý học Hoa Kỳ (APA), "Stress in America: Stress and Mental Health," 2021. [Online]. Available: https://www.apa.org/news/press/releases/stress. |
| [4] | "Báo cáo về sức khỏe tâm lý sinh viên trong môi trường học đường," Tạp chí Y tế công cộng Việt Nam, 2021. |
| [5] | "Student Depression," Kaggle Competition, [Online]. Available: https://www.kaggle.com/datasets/hopesb/student-depression-dataset/data. |
| [6] | Vũ Hữu Tiệp, "Machine Learning cơ bản," 2017. [Online]. Available: https://machinelearningcoban.com/. |
| [7] | Tạ Quang Chiểu, Slide bài giảng Học máy, Đại học Thuỷ Lợi. |

PHỤ LỤC

*##khai báo thư viện*

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import *StandardScaler*

from sklearn.preprocessing import *OneHotEncoder*

from sklearn.model\_selection import *train\_test\_split*

from sklearn.linear\_model import *LogisticRegression*

from sklearn.metrics import *confusion\_matrix*

import warnings

warnings.*filterwarnings*('ignore', *category*=*FutureWarning*)

*##đọc dữ liệu*

main = pd.*read\_csv*(*r*'Student Depression Dataset.csv')

*##tiền xử lý dữ liệu*

main = main.*drop*(['id'], *axis*=1)

main.*loc*[*main*['*Gender*'] *==* '*Male*','*Gender*'] = 0

main.*loc*[*main*['*Gender*'] *==* '*Female*','*Gender*'] = 1

main = main.*loc*[*main*['*Profession*'] *==* '*Student*']

main = main.drop(['Profession'], *axis*=1)

main = main.drop(['Work Pressure'], *axis*=1)

main = main.*loc*[*main*['*Age*']<= *30*]

main = main.*loc*[*main*['*Academic Pressure*']> *0*]

main = main.*loc*[*main*['*Study Satisfaction*']> *0*]

main = main.drop(['Job Satisfaction'], *axis*=1)

main = main.*loc*[*main*['*Sleep Duration*']!='*Others*']

main.*loc*[*main*['*Sleep Duration*']=='*Less than 5 hours*','*Sleep Duration*'] = 0

main.*loc*[*main*['*Sleep Duration*']=='*5-6 hours*','*Sleep Duration*'] = 1

main.*loc*[*main*['*Sleep Duration*']=='*7-8 hours*','*Sleep Duration*'] = 2

main.*loc*[*main*['*Sleep Duration*']=='*More than 8 hours*','*Sleep Duration*'] = 3

main = main.*loc*[*main*['*Dietary Habits*']!='*Others*']

main.*loc*[*main*['*Dietary Habits*']=='*Healthy*','*Dietary Habits*'] = 0

main.*loc*[*main*['*Dietary Habits*']=='*Unhealthy*','*Dietary Habits*'] = 1

main.*loc*[*main*['*Dietary Habits*']=='*Moderate*','*Dietary Habits*'] = 2

main.*loc*[*main*['*Degree*'].*str*.*contains*(*r*'*BSc*|*BCA*|*B.Ed*|*BHM*|*B.Pharm*|*B.Com*|*BE*|*BA*|*B.Arch*|*B.Tech*|*BBA*|*LLB*', *regex*=*True*),'*New\_Degree*'] = 'Graduated'

main.*loc*[*main*['*Degree*'].*str*.*contains*(*r*'*MSc*|*MCA*|*M.Ed*|*M.Pharm*|*M.Com*|*ME*|*MA*|*M.Arch*|*M.Tech*|*MBA*|*LLM*', *regex*=*True*),'*New\_Degree*'] = 'Post Graduated'

main.*loc*[*main*['*Degree*']=='*Class 12*','*New\_Degree*'] = 'Higher Secondary'

main = main.*loc*[*main*['*Degree*']!='*Others*']

main.*loc*[*main*['*New\_Degree*']=='*Graduated*','*New\_Degree*'] = 0

main.*loc*[*main*['*New\_Degree*']=='*Post Graduated*','*New\_Degree*'] = 1

main.*loc*[*main*['*New\_Degree*']=='*Higher Secondary*','*New\_Degree*'] = 2

main.*loc*[*main*['*Have you ever had suicidal thoughts ?*']=='*Yes*','*Have you ever had suicidal thoughts ?*'] = 1

main.*loc*[*main*['*Have you ever had suicidal thoughts ?*']=='*No*','*Have you ever had suicidal thoughts ?*'] = 0

main.*loc*[*main*['*Family History of Mental Illness*']=='*Yes*','*Family History of Mental Illness*'] = 1

main.*loc*[*main*['*Family History of Mental Illness*']=='*No*','*Family History of Mental Illness*'] = 0

main = main.drop(['City'], *axis*=1)

main = main.drop(['Degree'], *axis*=1)

main.isnull().sum()

main = main.dropna()

*##ma trận tương quan*

plt.*figure*(*figsize*=(20, 10))

sns.*heatmap*(main\_data.corr(), *annot*=True, *cmap*='coolwarm')

plt.*show*()

*##xây dựng mô hình*

X = main\_data.drop('Depression', *axis*=1).values

y = *main\_data*['*Depression*'].values

X\_train, X\_test, y\_train, y\_test = *train\_test\_split*(X, y, *test\_size*=0.2, *random\_state*=42)

scaler = *StandardScaler*()

X\_train\_scaled = scaler.*fit\_transform*(X\_train)

X\_test\_scaled = scaler.*transform*(X\_test)

model = *LogisticRegression*()

model.*fit*(X\_train\_scaled, y\_train)

score = model.*score*(X\_test\_scaled, y\_test)

*print*(*f*"Accuracy: {score\*100:.2f}%")

*##ma trận nhầm lẫn*

y\_pred = model.*predict*(X\_test\_scaled)

cm = *confusion\_matrix*(y\_test, y\_pred)

plt.*figure*(*figsize*=(6, 4))

sns.*heatmap*(cm, *annot*=True, *fmt*='d', *cmap*='Blues', *xticklabels*=['Predicted: 0', 'Predicted: 1'], *yticklabels*=['Real: 0', 'Real: 1'])

plt.*title*('Confusion Matrix')

plt.*xlabel*('Predicted Values')

plt.*ylabel*('Real Values')

plt.*show*()

*##so sánh các mô hình*

from sklearn.tree import *DecisionTreeClassifier*

from sklearn.svm import *SVC*

import xgboost as xgb

import lightgbm as lgb

from sklearn.metrics import *accuracy\_score*

models = {

    "Logistic Regression": *LogisticRegression*(),

    "Decision Tree": *DecisionTreeClassifier*(*random\_state*=42),

    "SVM": *SVC*(*random\_state*=42),

}

accuracy\_results = {}

for name, model in models.*items*():

    model.fit(X\_train\_scaled, y\_train)

    y\_pred = model.predict(X\_test\_scaled)

    accuracy = *accuracy\_score*(y\_test, y\_pred)

*accuracy\_results*[*name*] = accuracy

accuracy\_results\_ordened = *dict*(*sorted*(accuracy\_results.*items*(), *key*=lambda *item*: *item*[*1*], *reverse*=True))

plt.*figure*(*figsize*=(15, 6))

sns.*barplot*(*x*=*list*(accuracy\_results\_ordened.*values*()),

*y*=*list*(accuracy\_results\_ordened.*keys*()),

*palette*='viridis')

plt.*xlabel*('Accuracy')

plt.*title*('Model Accuracy')

for i, v in *enumerate*(accuracy\_results\_ordened.*values*()):

    plt.*text*(v, i, *f*'{v\*100:.2f}%', *color*='black', *va*='center')

plt.*show*()

*##k-fold*

from sklearn.model\_selection import *KFold*, *cross\_val\_score*

from sklearn.linear\_model import *LogisticRegression*

from sklearn.tree import *DecisionTreeClassifier*

from sklearn.svm import *SVC*

from sklearn.preprocessing import *StandardScaler*

from sklearn.metrics import *make\_scorer*, *accuracy\_score*

import numpy as np

import matplotlib.pyplot as plt

scaler = *StandardScaler*()

X\_scaled = scaler.*fit\_transform*(X)

models = {

    "Logistic Regression": *LogisticRegression*(*random\_state*=42),

    "Decision Tree": *DecisionTreeClassifier*(*random\_state*=42),

    "SVM": *SVC*(*random\_state*=42)

}

kf = *KFold*(*n\_splits*=5, *shuffle*=True, *random\_state*=42)

results = {name: [] for name in models.*keys*()}

training\_sizes = np.linspace(0.1, 0.9, 9)

for size in training\_sizes:

    X\_partial, \_, y\_partial, \_ = *train\_test\_split*(X\_scaled, y, *train\_size*=size, *random\_state*=42)

    for name, model in models.*items*():

        scores = *cross\_val\_score*(model, X\_partial, y\_partial, *cv*=kf, *scoring*=*make\_scorer*(*accuracy\_score*))

*results*[*name*].*append*(scores.mean())

*# Vẽ biểu đồ*

plt.*figure*(*figsize*=(10, 6))

for name, accuracies in results.*items*():

    plt.*plot*(training\_sizes, accuracies, *marker*='o', *label*=name)

plt.*title*("Model Performance vs Training Size with K-Fold", *fontsize*=14)

plt.*xlabel*("Training Size (Fraction of Training Data)", *fontsize*=12)

plt.*ylabel*("Mean Accuracy (K-Fold)", *fontsize*=12)

plt.*xticks*(training\_sizes, *labels*=[*f*"{*int*(size\*100)}%" for size in training\_sizes], *rotation*=45)

plt.*ylim*(0.5, 1.0)  *# Giới hạn trục tung từ 50% đến 100%*

plt.*legend*()

plt.*grid*(True, *linestyle*='--', *alpha*=0.7)

plt.*tight\_layout*()

plt.*show*()

*##triển khai mô hình*

import pandas as pd

import numpy as np

from sklearn.linear\_model import *LogisticRegression*

from sklearn.model\_selection import *train\_test\_split*

from sklearn.metrics import *accuracy\_score*, *classification\_report*

column\_names = ['Gender', 'Age', 'Academic Pressure',

                'CGPA', 'Study Satisfaction',

                'Sleep Duration', 'Dietary Habits', 'New\_Degree',

                'Have you ever had suicidal thoughts ?', 'Work/Study Hours',

                'Financial Stress', 'Family History of Mental Illness']

X\_train = np.random.rand(100, 12)

y\_train = np.random.randint(0, 2, *size*=100)

logistic\_model = *LogisticRegression*(*max\_iter*=1000)

logistic\_model.*fit*(X\_train, y\_train)

def *predict\_depression*(*user\_input*):

    """

    Predicts depression risk based on user input.

    Parameters:

        user\_input (dict): Dictionary containing user-provided information.

    Returns:

        str: Prediction result ('HAS SIGN OF DEPRESSION' or 'HAS NO SIGN OF DEPRESSION').

    """

    input\_df = pd.*DataFrame*([*user\_input*])

    input\_df = input\_df.*reindex*(*columns*=column\_names, *fill\_value*=0)

    prediction = logistic\_model.*predict*(input\_df)[0]

    return 'HAS SIGN OF DEPRESSION' if prediction == 1 else 'HAS NO SIGN OF DEPRESSION'

example\_input = {

    'Gender': *int*(*input*("Gender (0 for Male, 1 for Female): ")),

    'Age': *int*(*input*("Age (18-30): ")),

    'Academic Pressure': *int*(*input*("Academic Pressure (0-5): ")),

    'CGPA': *float*(*input*("CGPA (0-10): ")),

    'Study Satisfaction': *int*(*input*("Study Satisfaction (0-5): ")),

    'Sleep Duration': *int*(*input*("Sleep Duration (0 for less than 5 hours, 1 for 5-6 hours, 2 for 7-8 hours, 3 for more than 8 hours): ")),

    'Dietary Habits': *int*(*input*("Dietary Habits (0 for healthy, 1 for unhealthy, 2 for moderate): ")),

    'New\_Degree': *int*(*input*("Degree (0 for Graduated, 1 for Post Graduated, 2 for Higher Secondary): ")),

    'Have you ever had suicidal thoughts ?': *int*(*input*("Have you ever had suicidal thoughts ? (0 for No, 1 for Yes): ")),

    'Work/Study Hours': *int*(*input*("Work/Study Hours per day: ")),

    'Financial Stress': *int*(*input*("Financial Stress (0-5): ")),

    'Family History of Mental Illness': *int*(*input*("Family History of Mental Illness (0 for No, 1 for Yes): "))

}

result = *predict\_depression*(example\_input)

*print*(*f*">>>RESULT: {result}")