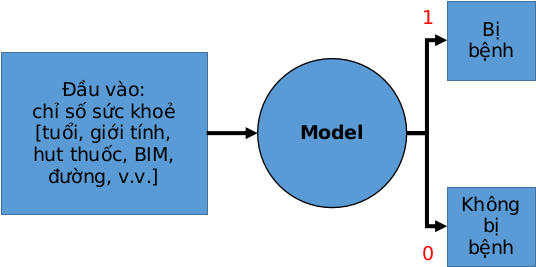
Logistic Regression

1. Giới thiệu

Hồi quy logistic là một thuật toán học máy có giám sát được sử dụng cho các vấn đề phân loại. Không giống như hồi quy tuyến tính dự đoán các giá trị liên tục, nó dự đoán xác suất đầu vào thuộc về một lớp cụ thể. Nó được sử dụng để phân loại nhị phân, trong đó đầu ra có thể là một trong hai loại có thể có như Có/Không, Đúng/Sai hoặc 0/1. Nó sử dụng hàm sigmoid để chuyển đổi đầu vào thành giá trị xác suất giữa 0 và 1. Hãy xem những điều cơ bản về hồi quy logistic và các khái niệm cốt lõi của nó.

1. Cách hoạt động

ví dụ: Hãy xem xét vấn đề phát hiện xem một người có bị bệnh tim hay không bị bệnh tim.



Đầu vào có thể biểu diễn như vector ], với mỗi thành phần tương ứng với một đặc trưng trong một mẫu bệnh án.

- Để có mô hình đáp ứng được điều này ta thử quay lại bài toán hồi quy tuyến tính .

**Model**

]

x = [1, 2, 1.2, 2.1, 1.4]

Đại điện một sample

x = [1, 2, 1.2, 2.1, 5]

giả sử

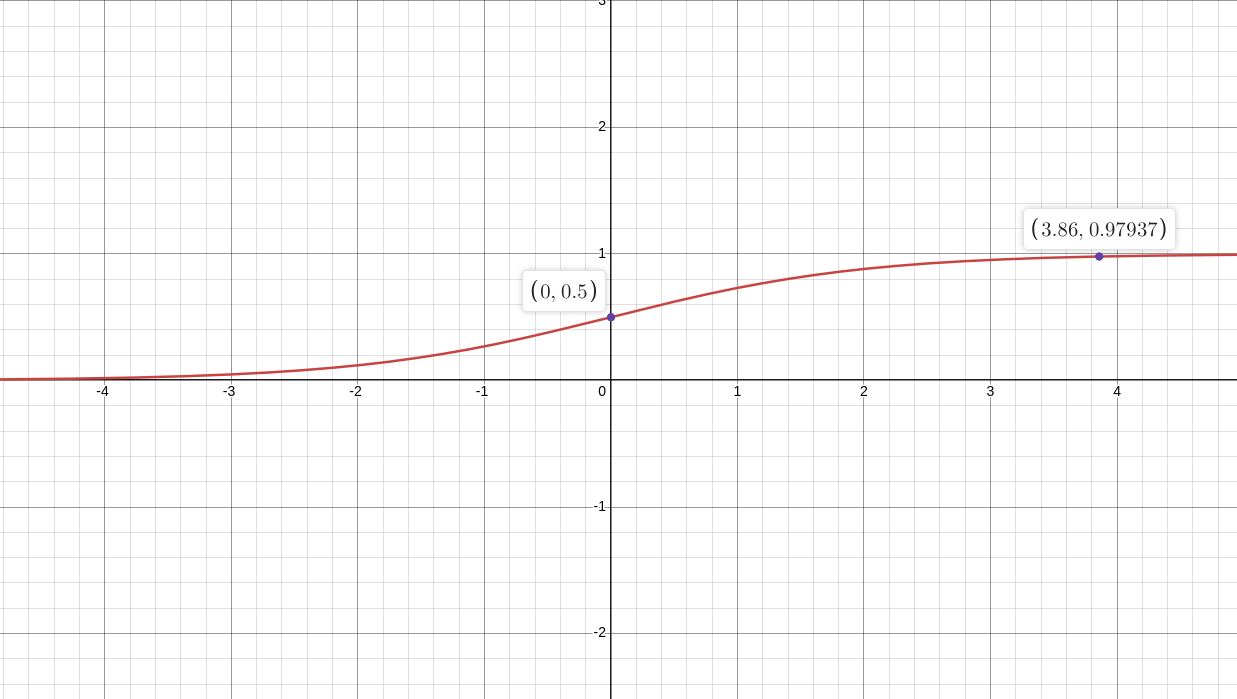
=3.86

giả sử b=1

vậy làm cách nào để kết quả đầu ra có giá trị rời rạc?

1. Giải pháp

Giải pháp cho vấn đề trên ta có hàm Sigmoid đây là một giải pháp để tạo ra giá trị rời rạc, xem đồ thị của hàm Sigmoid.



- Xét một ví dụ xác suất thống kê cho việc tung đồng xu. Tung đồng xu 5 lần được 3 lần mặt ngửa (đặt là 1) và 2 lần mặt sấp (đặt là 0).

Xác suất mặt ngửa: P(y=1|x)=.

Xác suất mặt sấp : P(y=0|x)=.

Vậy xác suất:

trong đó là xác xuất dự đoán y=1.

viết gọn lại: P(y|x)=, y=0 hoặc y=1.

Để dễ tính ta lấy log hai vế.

log(P(y|x))=)

=)+log()

=)+).log() (\*)

Mục tiêu của bài toán là tối đa hoá dự đoán càng gần nhãn thực tế thì càng tốt: nhãn thực tế y=1, dự đoán =0.999 kết luận tốt nhưng nếu nhãn thực tế y=1, dự đoán =0.001 kết luận xấu.

- giờ hãy xem ý nghĩa thự tế của biểu thức (\*).

nếu y=1, (\*)<=> );

|  |  |  |  |
| --- | --- | --- | --- |
| y  (thực tế) | (dự đoán) | ) | ý nghĩa |
| 1 | 0.999 | -0.0004345 | tốt ( gần 0) |
| 1 | 0.9 | -0.0457574 | khá tốt |
| 1 | 0.4 | -0.39794 | không tốt |
| 1 | 0.001 | -3 | tệ |

kết luận: khi mô hình dự đoán gần 1 thì gần 0 => mô hình tốt, ngược lại gần 0 thì âm rất lớn => mô hình tệ.

nếu y=0, (\*)<=> )

|  |  |  |  |
| --- | --- | --- | --- |
| y  (thực tế) | (dự đoán) | ) | ý nghĩa |
| 0 | 0.999 | -3 | tệ |
| 0 | 0.9 | -1 | không tốt |
| 0 | 0.4 | -0.221848 | trung bình |
| 0 | 0.001 | -0.0004345 | tốt ( gần 0) |

kết luận: khi mô hình dự đoán gần 1 thì 1-gần 0 và (\*)

âm rất lớn => mô hình tệ, ngược lại gần 0 thì 1-gần 1 và gần 0 => mô hình tốt.

từ đây ta có thể suy ra được hàm mất mát dựa vào (\*)

Loss=-=- [)+).log()]

ta cần lấy dấu âm vế phải của biểu thức (\*):

+ mô hình dự đoán tốt: gần 0=>Loss nhỏ(gần 0)

+ mô hình dự đoán tệ: âm lớn=>Loss rất lớn

(vd: Loss=-=-(-3)=3)

Vậy tiếp theo ta cần phải làm gì khi đã có được hàm Loss.

Để có được mô hình dự đoán tốt thì ta cần cực tiểu hoá hàm Loss

1. Thuật toán

Trong tập huấn luyện có n mẫu.

L(w,b) =-]

trong đó và

- đạo hàm riêng Loss với w.

* =+=
* ==()=

=>=

suy ra ()=(1-)

* ===

suy ra

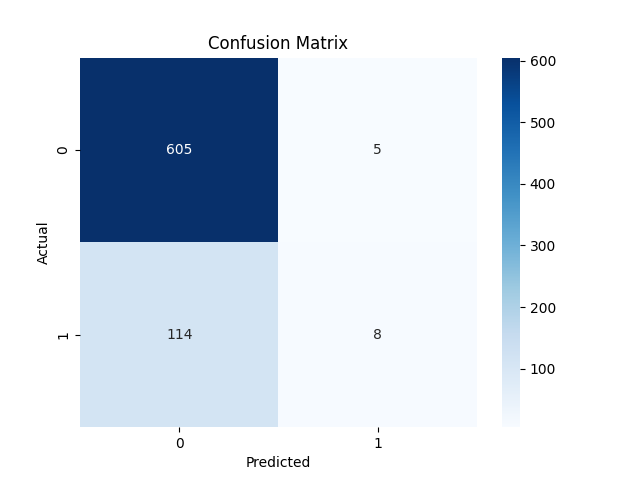
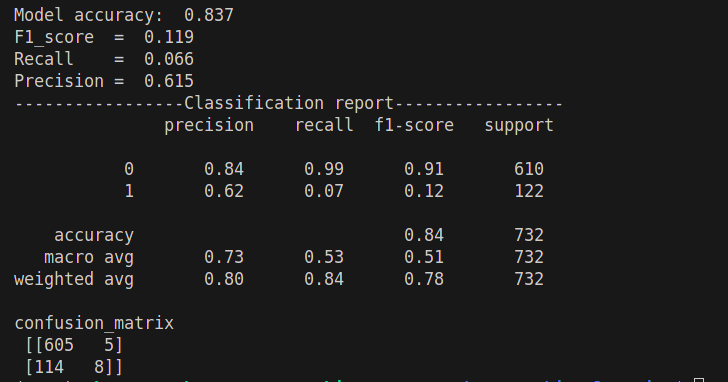
=

=

- đạo hàm riêng Loss với w.

* ===1

1. Kiểm thử



chạy thử thuật toán

Nhận thấy accuracy của mô hình khá cao tuy nhiên mục tiêu dự đoán những người bị bệnh thật sự rất kém Recall= 7%

-thấy rằng Model accuracy: 0.837 dễ nhầm lẫn thành model dự đoán tốt tuy nhiên Recall= 0.066 lại rất thấp

->dẫn đến việc dự đoán người thực sự mắc bệnh thì model dự đoán lại rất tệ nhìn vào confusion\_matrix có đến

122 người thực sự mắc bệnh mà chỉ dự đoán được 8 người

**Nguyên nhân**

do bộ dữ liệu bị mát cân bằng giữa class bị bệnh (y=1) và class không bị bệnh (y=0), nên mô hình tập trung

vào học các đặc trưng của các samples không bị bệnh (y=0).

**Giải pháp**

có 2 cách điển hình.

- cách 1: Cân bằng dữ liệu huấn luyện nhân bản dữ liệu những class bị bệnh để model có thể học được

nhiều hơn về class bị bệnh(y=1).

+kết quả:

Before SMOTE:

Number of samples in each class: Counter({0: 2489, 1: 435})

After SMOTE:

Number of samples in each class: Counter({0: 2489, 1: 2489})

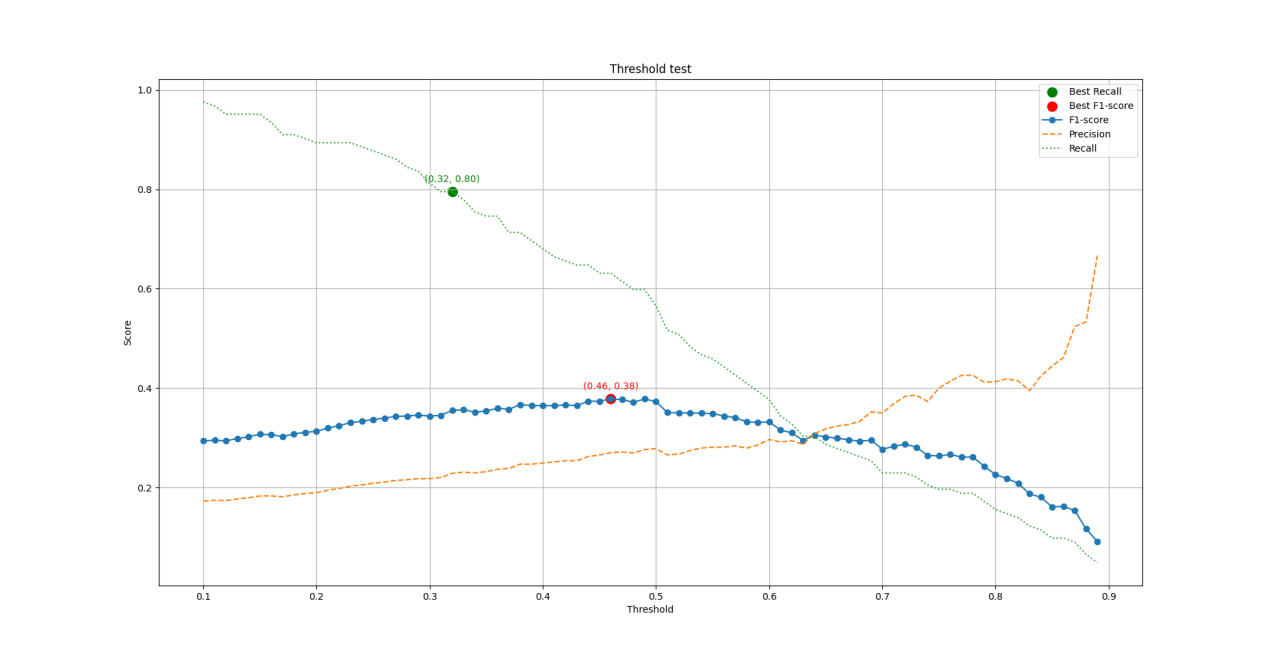
- cách 2: Ta sẽ cần tìm điểm cân bằng sao cho model có thể dự đoán được nhiều người bị bệnh thật sự

nhưng cũng không được cảnh báo nhầm nhiều người từ không bị bệnh thành bị bệnh. bằng cách tìm điểm ngưỡng

để cân bằng F1\_score. tuy nhiên nếu muốn model không bỏ sót người bị bệnh thực sự thì ta lại phải đánh đổi

việc model cảnh báo nhầm những người không bị bệnh nhưng lại được dự đoán là bị bệnh.

**Tìm điểm tối ưu.**



Điểm tối ưu cho F1 và Recall

Cho mô hình thay đổi ngưỡng từ 0.1 đến 0.9 để tìm ngưỡng tối ưu cho 2 metrics F1 score và Recall.

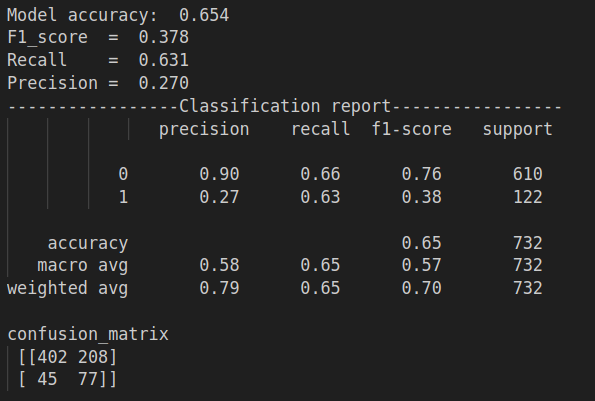
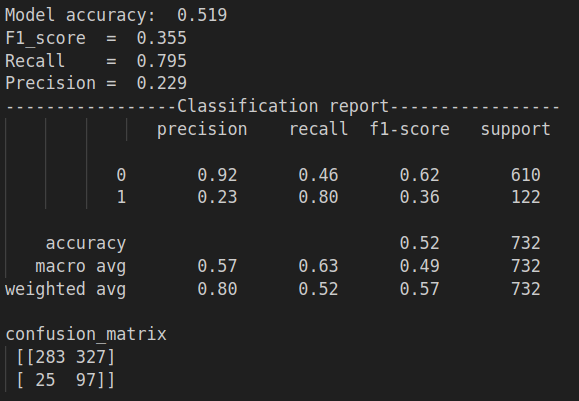
+kết quả: khi áp dụng cả cách 1 và 2 ta tìm được điểm ngưỡng tốt nhất như sau.

Threshold best F1-score: 0.46 with F1 = 0.378

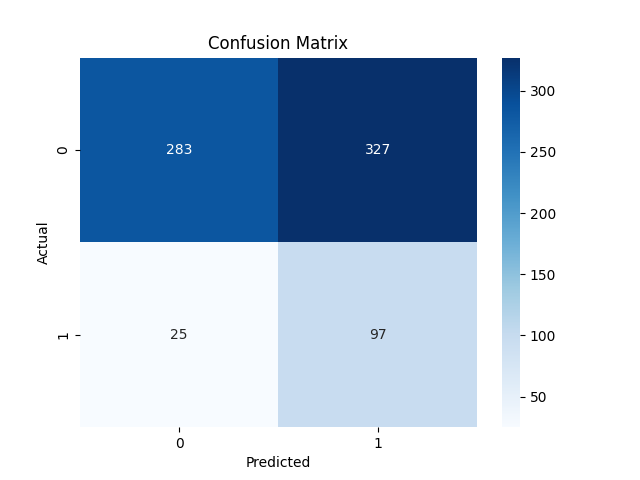
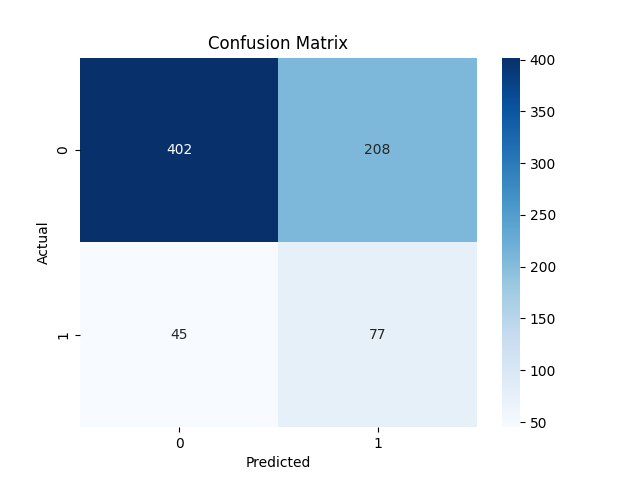
Threshold best Recall: 0.32 with Recall = 0.795

**Tinh chỉnh**

Điều chỉnh tham số mô hình cụ thể là 2 ngưỡng mới tìm được ở trên để xem xét mô hình



Best F1 Best Recall



Best F1 confusion matrix Best Recall confusion matrix