Báo cáo đồ án cuối khóa

Bài toán phân tích dự đoán về tỉ lệ bệnh nhân tử vong do mắc bệnh suy tim

Contents

1. Giới thiệu về bộ dữ liệu và mục đích xây dựng mô hình để giải quyết vấn đề bài toán

- Suy tim là một biến chứng phổ biến do CVD (cardiovascular diseases) gây ra và là 1 trong những nguyên nhân gây tử vong số 1 trên toàn cầu, ước tính cướp đi sinh mạng của 17,9 triệu người mỗi năm, chiếm 31% tổng số ca tử vong trên toàn thế giới. Bộ dữ liệu chứa 12 đặc điểm có thể được sử dụng để dự đoán tỷ lệ tử vong do suy tim. Hầu hết các bệnh tim mạch có thể được ngăn ngừa bằng cách giải quyết các yếu tố nguy cơ hành vi như sử dụng thuốc lá, chế độ ăn uống không lành mạnh và béo phì, lười vận động và sử dụng rượu có hại bằng nhiều phương pháp khác nhau.
- Bộ dữ liệu chứa các thông tin lâm sàng về bệnh nhân bao gồm các thuộc tính: age, anaemia, creatinine_phosphokinase, diabetes, ejection_fraction, high_blood_pressure, platelets, serum_creatinine, serum_sodium, sex, smoking, time và DEATH_EVENT (biến mục tiêu tỉ lệ tử vong). Dữ liệu được thu thập từ các bệnh nhân bị suy tim và đã được gán nhãn để xác định liệu họ có tử vong trong khoảng thời gian quan sát hay không.
- Bệnh suy tim là một trong những vấn đề sức khỏe nghiêm trọng đang diễn ra trên toàn thế giới và ảnh hưởng đến hàng triệu người. Mục tiêu của project này là xây dựng một mô hình dự đoán tỉ lệ tử vong do suy tim dựa trên các thông tin lâm sàng và các yếu tố

liên quan. Chúng ta sẽ sử dụng các mô hình thuật toán để phân tích dữ liệu và dự đoán xác suất tử vong dự kiến của bệnh nhân. Từ đó có thể giúp cho các bác sĩ và chuyên gia y tế có cái nhìn sâu hơn về tình trạng sức khỏe của bệnh nhân và đưa ra quyết định điều trị phù hợp và kịp thời.

2. Các phương pháp data processing và kết quả

- Sử dụng dataframe đọc file dữ liệu csv

```
df = pd.read_csv("./Heart_failure_clinical_records_dataset.csv")
```
✓ 0.0s Python

```
df
```
✓ 0.0s Python

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	4
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	6
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	7
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	7
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	8
...	...	...	...	...	...	...	...	...	...	...	...	...
294	62.0	0	61	1	38	1	155000.00	1.1	143	1	1	270
295	55.0	0	1820	0	38	0	270000.00	1.2	139	0	0	271
296	45.0	0	2060	1	60	0	742000.00	0.8	138	0	0	278
297	45.0	0	2413	0	38	0	140000.00	1.4	140	1	1	280
298	50.0	0	196	0	45	0	395000.00	1.6	136	1	1	285

299 rows × 13 columns

- Lấy số lượng cột và dòng, đồng thời lấy tên các cột

```
# Get the number of rows and columns
print(df.shape)
# get the column names
print(df.columns)
```

✓ 0.0s

```
(299, 13)
Index(['age', 'anaemia', 'creatinine_phosphokinase', 'diabetes',
       'ejection_fraction', 'high_blood_pressure', 'platelets',
       'serum_creatinine', 'serum_sodium', 'sex', 'smoking', 'time',
       'DEATH_EVENT'],
      dtype='object')
```

- Tìm các kí tự đặc biệt trong giá trị từng cột và dòng

```
for i in df.columns:
    if is_numeric_dtype(df[i]) == False:
        list_char = []
        for j in range(len(df)):
            if type(df[i][j]) == str:
                list_char.extend(re.findall("[^A-Za-z0-9]", df[i][j]))
        print(i, list(dict.fromkeys(list_char)))
```

✓ 0.0s

- Tìm và xóa các giá trị trùng lặp

```
# show duplicate rows
df[df.duplicated()]
```

✓ 0.0s Python

age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEAT

```
# Dropping the duplicate Rows
print("Initial shape of the dataset : ", df.shape)
df = df.drop_duplicates(keep = 'first', ignore_index=True)
print ("Shape of the dataset after dropping the duplicate rows : ", df.shape)
```

✓ 0.0s Python

```
Initial shape of the dataset :  (299, 13)
Shape of the dataset after dropping the duplicate rows :  (299, 13)
```

- Tìm và đếm số lượng giá trị riêng biệt trong tập dữ liệu

```
# Loop through each column and count the number of distinct values
for column in df.columns:
    num_distinct_values = len(df[column].unique())
    print(f"{column} -> {num_distinct_values} distinct values\n")
```

✓ 0.0s

```
platelets -> 176 distinct values

serum_creatinine -> 40 distinct values

serum_sodium -> 27 distinct values

sex -> 2 distinct values

smoking -> 2 distinct values
```

- Do tất cả dữ liệu trong tập dữ liệu đều ở dạng số hóa nên không cần thêm bước số hóa dữ liệu

3. Các thuật toán sử dụng

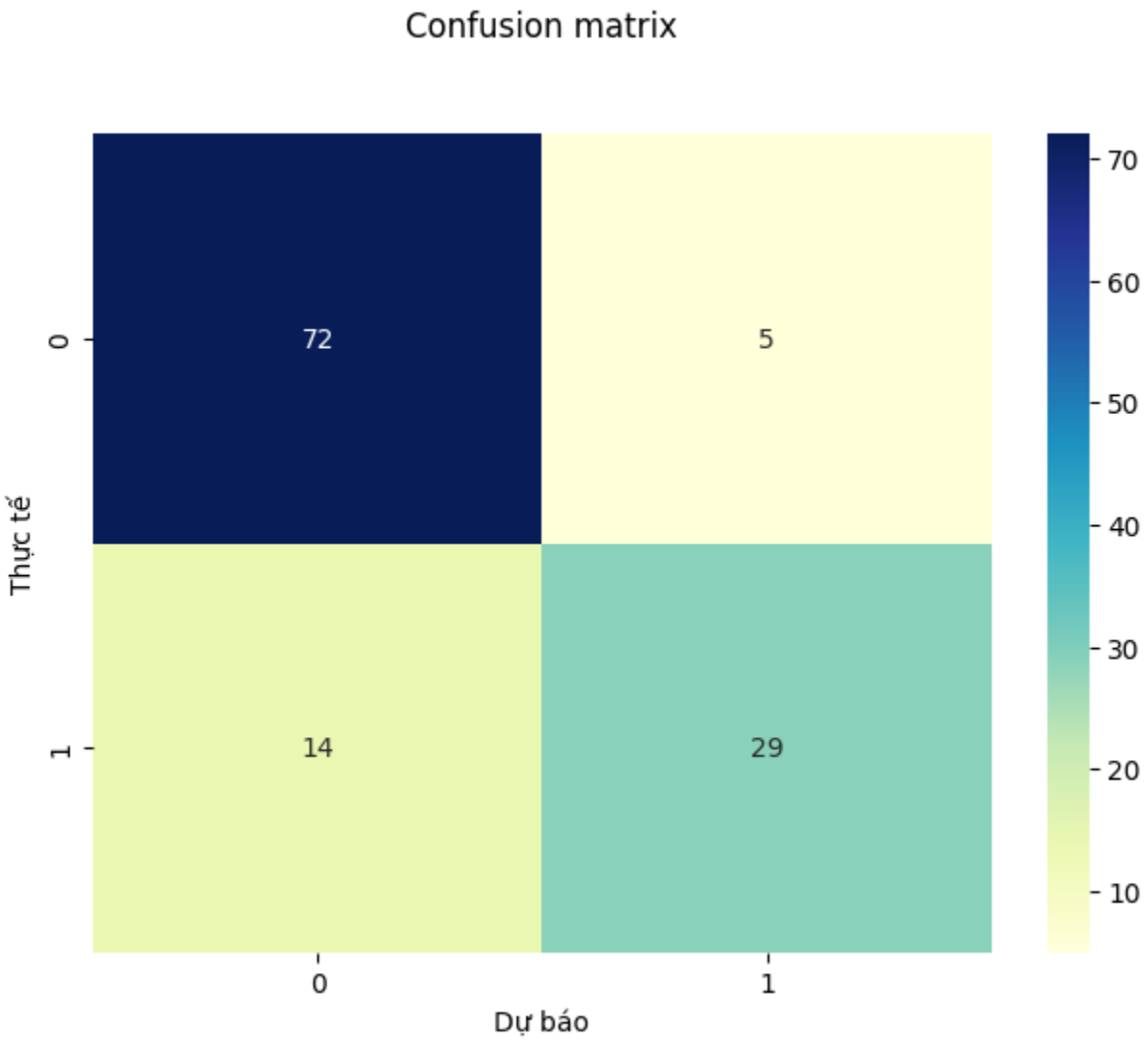
Chúng ta sẽ sử dụng biến DEATH_EVENT trong bộ dữ liệu để làm biến target và một loạt các thuật toán phân loại để dự đoán tỉ lệ tử vong, bao gồm Logistic Regression và Support Vector Machine.

- Logistic Regression
  - Chia các cột vào giá trị x và y (y là cột DEATH_EVENT) và chia tập dữ liệu thành train_size 70 và test_size 30
  - Khởi tạo mô hình Logistic Regression và huấn luyện dữ liệu để mô hình học từ dữ liệu và dự đoán, tính giá trị accuracy của mô hình thuật toán
  - Sử dụng thư viện Seaborn và scikit-learn để tính toán và hiển thị ma trận nhầm lẫn (confusion matrix) của mô hình Logistic Regression đã huấn luyện trước đó trên dữ liệu kiểm tra, sau đó tạo một heatmap (biểu đồ màu) cho ma trận nhầm lẫn (confusion matrix) đã tính toán trước đó và hiển thị các thông tin quan trọng liên quan đến hiệu suất của mô hình Logistic Regression trên dữ liệu kiểm tra.
  - Tính toán và in ra Classification Report gồm precision, recall, f1-score, support
- Support Vector Machine

- Khởi tạo mô hình SVC cho SVM và thiết lập hàm kernel, tham số C và tham số Gamma. Hàm kernel là linear, tức là SVM sẽ thực hiện phân loại tuyến tính. Giá trị C lớn hơn (C=10) cũng có nghĩa là mô hình sẽ cố gắng phân loại đúng càng nhiều điểm dữ liệu trong tập huấn luyện mặc dù có thể phải chấp nhận sai số phân loại. Cuối cùng là huấn luyện dữ liệu
- Tính toán và đưa ra accuracy của thuật toán
- Đưa ra Classification Report và Confusion Matrix

4. Kết quả và so sánh

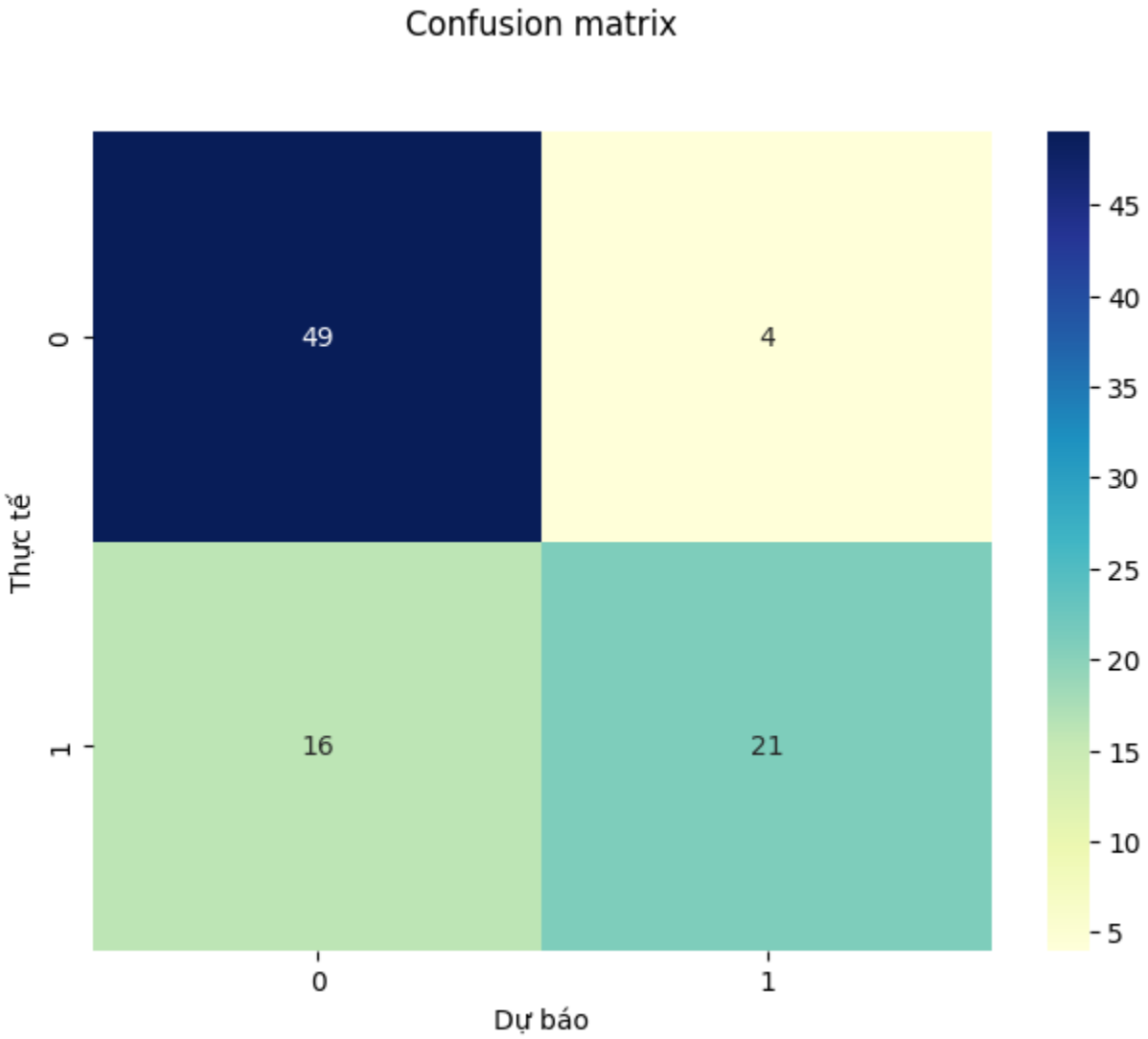
- Logistic Regression
  - Confusion Matrix



- Classification Report

	precision	recall	f1-score	support
0	0.84	0.94	0.88	77
1	0.85	0.67	0.75	43
accuracy			0.84	120
macro avg	0.85	0.80	0.82	120
weighted avg	0.84	0.84	0.84	120

- Support Vector Machine
  - Confusion Matrix



- Classification Report

	precision	recall	f1-score	support
0	0.75	0.92	0.83	53
1	0.84	0.57	0.68	37
accuracy			0.78	90
macro avg	0.80	0.75	0.75	90
weighted avg	0.79	0.78	0.77	90

Dựa vào kết quả trên, ta đều có thể nhận định được rằng cả 2 mô hình đều cho ra độ chính xác khá cao (0.84 cho Logistic và 0.78 cho SVM).

Đối với chỉ số Precision của cả 2 mô hình cho 2 trường hợp tử vong và không tử vong, cả 2 mô hình đều có khả năng dự đoán chính xác cao cho các ca sẽ tử vong và không tử vong (theo lý thuyết, precision càng cao thì khả năng dự đoán chính xác cho lớp 0 và lớp 1 sẽ càng cao).

Đối với chỉ số Recall của cả 2 mô hình, cả 2 đều cho chỉ số khá thấp, dẫn đến khả năng cả 2 mô hình sẽ bỏ qua 1 số ca tử vong.

Kết luận, cả 2 mô hình đều có khả năng dự đoán và phân loại khá tốt, cũng như độ chính xác cao cho các ca không tử vong (lớp 0), còn dự đoán về các ca tử vong cần phải được cải thiện hơn.