

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM

KHOA CÔNG NGHỆ THÔNG TIN



MAI ANH KHOA 21110836

HOÀNG CÔNG MẠNH 21110839

NGUYỄN TRƯỜNG AN 21110117

DỰ ĐOÁN KHẢ NĂNG MẮC AIDS

LỚP HỌC PHẦN: 21110CLC_MALE431085_04CLC

GIÁO VIÊN HƯỚNG DẪN

ThS. Trần Quang Khải

Thành phố Hồ Chí Minh, tháng 5 năm 2024

NHẬN XÉT CỦA GIẢNG VIÊN

LỜI CẢM ƠN

Chúng em xin gửi đến thầy lời cảm ơn chân thành nhất về sự hỗ trợ và sự chỉ dẫn tận tình mà cô đã dành cho chúng em trong suốt quá trình xây dựng Đề tài "Dự Đoán Khả Năng Mắc AIDS" cho môn học Học Máy.

Sự nhiệt tình và kiên nhẫn của thầy đã giúp chúng em vượt qua những thách thức và khó khăn trong quá trình thực hiện dự án. Những lời khuyên và phản hồi của cô đã giúp chúng em hiểu rõ hơn về các khái niệm và kỹ thuật trong các mô hình học máy, từ đó giúp chúng em hoàn thành dự án một cách thành công và hiệu quả.

Chúng em rất may mắn và tự hào khi được học và làm việc dưới sự hướng dẫn của một người giáo viên như thầy. Sự đóng góp của thầy không chỉ là một phần quan trọng trong dự án của chúng em mà còn là một nguồn động viên lớn lao để chúng em tiếp tục phát triển và trưởng thành trong hành trình học tập của mình.

Một lần nữa, chúng em xin chân thành cảm ơn thầy về tất cả những gì đã làm cho chúng em. Chúng em rất biết ơn và trân trọng.

MỤC LỤC

PHẦN MỞ ĐẦU	5
1. Lý do chọn đề tài.	5
2. Phương pháp thực hiện.	5
3. Mục tiêu của đề tài.	5
PHẦN NỘI DUNG	7
CHƯƠNG 1. GIỚI THIỆU TẬP DỮ LIỆU ĐÃ SỬ DỤNG	7
1.1. Mô tả	7
1.2. Các thuộc tính	7
CHƯƠNG 2. KHAI PHÁ DỮ LIỆU	9
2.1. Lấy dữ liệu.	9
2.2. Một vài giá trị đầu tiên	10
2.3. Mô tả dữ liệu	11
2.4. Kiểm tra dữ liệu trùng lặp	11
CHƯƠNG 3: XỬ LÝ CÁC THUỘC TÍNH	12
3.1. Thuộc tính định lượng	12
3.1.1. time:	12
3.1.2. age:	14
3.1.3. wtkg:	15
3.1.4. karnofsky:	17
3.1.5. preanti	19
3.1.6. cd40:	21
3.1.7. cd420	23
3.1.8. cd80:	24
3.1.9. cd820:	26
3.2. Thuộc tính phân loại	28
3.2.1. infected:	28
3.2.2. trt:	29
3.2.3. hemo:	29
3.2.4.homo:	30
3.2.5. drugs:	30
3.2.6. oprior:	31
3.2.7. z30:	31
3.2.8. race:	32
3.2.9. gender:	32
3.2.10. str2:	33
3.2.11. strat:	33
3.2.12. symptom:	34
Nhận xét:	34
• Đa số là những người không có triệu chứng. Trong tổng số những người có triệu chứng, tỉ lệ người nhiễm cao hơn (50%).	34
3.2.13. treat:	34
3.2.14. offtrt:	35
3.3. Biểu đồ tương quan	35

CHƯƠNG 4: HUẤN LUYỆN MÔ HÌNH DỰ ĐOÁN KHẢ NĂNG MẮC BỆNH AIDS	36
4.1. Chuẩn bị tập dữ liệu	36
4.2. Scale dữ liệu	37
4.3. Xây dựng mô hình	37
4.4. Cải thiện mô hình	39
4.4.1. Random Forest Classifier	40
4.4.2. Gradient Boost Classifier	46
4.4.3. XG Boost Classifier	51
PHẦN KẾT LUẬN	56
TÀI LIỆU THAM KHẢO	57

PHẦN MỞ ĐẦU

1. Lý do chọn đề tài.

Bệnh AIDS (acquired immunodeficiency syndrome - hội chứng suy giảm miễn dịch mắc phải), hay HIV/AIDS, là một trong những vấn đề sức khỏe toàn cầu nghiêm trọng, gây ra bởi virus HIV khi loại virus này tấn công vào hệ miễn dịch của cơ thể. Không chỉ gây ra những vấn đề sức khỏe nghiêm trọng như suy giảm miễn dịch và các bệnh liên quan, AIDS còn là một trong những nguyên nhân gây ra tử vong cao. Tuy đã có những tiến bộ trong việc điều trị và quản lý bệnh, nhưng vẫn còn nhiều thách thức, đặc biệt là việc phát hiện sớm và điều trị kịp thời.

Theo các thống kê từ tổ chức Y tế Thế giới (WHO), dù đã có những nỗ lực để kiểm soát bệnh AIDS, nhưng vẫn có hàng triệu người mắc bệnh mỗi năm, và tình trạng này đặc biệt nghiêm trọng ở những khu vực có tài nguyên y tế hạn chế. Điều này thúc đẩy nhu cầu can thiệp và điều trị kịp thời, từ đó giảm thiểu sự lan truyền của virus và cải thiện chất lượng cuộc sống của những người bị ảnh hưởng. Điều này thể hiện sự cần thiết và tính cấp bách của việc nghiên cứu và áp dụng các phương pháp dự đoán AIDS trong việc quản lý và kiểm soát dịch bệnh. Vì vậy nhóm chúng em quyết định lựa chọn đề tài ***“Dự Đoán Khả Năng Mắc AIDS”*** làm đề tài cho môn học này.

2. Phương pháp thực hiện.

- Phân tích nhu cầu thực tế.
- Tìm kiếm tập dữ liệu phù hợp.
- Áp dụng các phương pháp học máy vào xử lý tập dữ liệu.

3. Mục tiêu của đề tài.

Xây dựng tập dữ liệu học máy dự đoán chính xác khả năng mắc bệnh AIDS.

PHẦN NỘI DUNG

CHƯƠNG 1. GIỚI THIỆU TẬP DỮ LIỆU ĐÃ SỬ DỤNG

1.1. Mô tả

- Tập dữ liệu: **AIDS Virus Infection Prediction**
- Tập dữ liệu này bao gồm các thống kê y tế và các thông tin phân loại về những bệnh nhân đã được chẩn đoán nhiễm AIDS. Tập dữ liệu này được đăng vào năm 1996.
- Tập dữ liệu bao gồm nhiều file csv khác nhau, có những số lượng record khác nhau (từ 2000-50000 record) và để đảm bảo khả năng dự đoán, chúng ta sẽ dùng tập dữ liệu cao nhất là 50000 record.

1.2. Các thuộc tính

STT	Tên	Ý nghĩa
1	time	Thời gian thực hiện nghiên cứu cho đến khi bị suy gan (theo ngày)
2	trt	Liệu pháp điều trị (0: chỉ ZDV, 1: ZDV + ddl, 2: ZDV + Zai, 3: chỉ ddl)
3	age	Tuổi tính theo năm
4	wtkg	Cân nặng theo kg
5	hemo	Mắc bệnh giảm đông máu (hemophila) (0: false, 1: true)
6	homo	Hoạt động đồng tính (0: false, 1: true)
7	drugs	Lịch sử dùng ma túy theo đường tĩnh mạch (IV) (0: false, 1: true)
8	karnof	Điểm số Karnofsky dùng để đánh giá chất lượng cuộc sống (từ 0: qua đời, đến 100: chất lượng sống cao nhất)
9	oprior	Đã dùng thuốc chống ức chế virus không ZDV hay chưa (0: false, 1: true)
10	z30	Sử dụng zdv trong 30 ngày. (0: false, 1: true)
11	preanti	Số ngày dùng liệu pháp chống ức chế nhân lên của virus (antiretroviral)
12	race	Sắc tộc (0: trắng, 1: còn lại)

13	gender	Giới tính (0: F, 1: M)
14	str2	Lịch sử dùng antiretroviral (0: chưa, 1: đã trải qua)
15	strat	Phân loại lịch sử dùng antiretroviral (1: chưa dùng, 2: đã dùng thuốc trong từ 1 đến 52 tuần, 3: đã dùng hơn 52 tuần)
16	symptom	Có triệu chứng bệnh không (0: asymp - không có, 1: symp - có)
17	treat	Liệu pháp điều trị (0: chỉ ZDV, 1: khác)
18	offtrt	Có dùng sử dụng trong 96+/- 5 tuần không (0: false, 1: true)
19	cd40	Số lượng tế bào miễn dịch CD4
20	cd420	Số lượng tế bào CD4 trong 20+/-5 tuần
21	cd80	Số lượng tế bào cd8
22	cd820	Số lượng tế bào CD8 trong 20+/-5 tuần
23	infected	Cho biết có nhiễm hay không (0: false, 1: true)

Bảng 2.1: Các thuộc tính của tập dữ liệu.

CHƯƠNG 2. KHAI PHÁ DỮ LIỆU

2.1. Lấy dữ liệu.

Công việc cần thiết khi khai phá dữ liệu là lấy được tập dữ liệu về. Trước tiên chúng ta hãy import những thư viện sau để phục vụ quá trình học máy:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.metrics import roc_curve, auc
from sklearn.decomposition import PCA
from imblearn.over_sampling import SMOTE
from sklearn.svm import SVC
```

- Code lấy tập dữ liệu về:

```
data = pd.read_csv('https://drive.google.com/uc?export=download&id=10nymCe3
Hl7w71R-0u3UZhRkLGcv2vJ0d')
df = data.copy()
df.info()
```

- Kết quả:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2139 entries, 0 to 2138
Data columns (total 23 columns):
#   Column      Non-Null Count  Dtype
---  -
0   time        2139 non-null   int64
1   trt         2139 non-null   int64
2   age         2139 non-null   int64
3   wtkg        2139 non-null   float64
4   hemo        2139 non-null   int64
5   homo        2139 non-null   int64
6   drugs       2139 non-null   int64
7   karnof      2139 non-null   int64
8   oprior      2139 non-null   int64
9   z30         2139 non-null   int64
10  preanti     2139 non-null   int64
11  race        2139 non-null   int64
12  gender      2139 non-null   int64
13  str2        2139 non-null   int64
14  strat       2139 non-null   int64
15  symptom     2139 non-null   int64
16  treat       2139 non-null   int64
17  offtrt      2139 non-null   int64
18  cd40        2139 non-null   int64
19  cd420       2139 non-null   int64
20  cd80        2139 non-null   int64
21  cd820       2139 non-null   int64
22  infected    2139 non-null   int64
dtypes: float64(1), int64(22)
memory usage: 384.5 KB
```

- Chúng ta dễ dàng thấy được không có dữ liệu null ở các đặc trưng.

```
[ ] df.shape
```

(2139, 23)


- Tập dữ liệu trên có 50000 mẫu và 23 đặc trưng


2.2. Một vài giá trị đầu tiên

```
[ ] df.head()
```

	time	trt	age	wtkg	hemo	homo	drugs	karnof	oprior	z30	...	str2	strat	symptom	treat	offtrt	cd40	cd420	cd80	cd820	infected
0	948	2	48	89.8128	0	0	0	100	0	0	...	0	1	0	1	0	422	477	566	324	0
1	1002	3	61	49.4424	0	0	0	90	0	1	...	1	3	0	1	0	162	218	392	564	1
2	961	3	45	88.4520	0	1	1	90	0	1	...	1	3	0	1	1	326	274	2063	1893	0
3	1166	3	47	85.2768	0	1	0	100	0	1	...	1	3	0	1	0	287	394	1590	966	0
4	1090	0	43	66.6792	0	1	0	100	0	1	...	1	3	0	0	0	504	353	870	782	0








2.3. Mô tả dữ liệu

 `df.describe()`




	time	trt	age	wtkg	hemo	homo	drugs	karnof	oprior	z30
count	2139.000000	2139.000000	2139.000000	2139.000000	2139.000000	2139.000000	2139.000000	2139.000000	2139.000000	2139.000000
mean	879.098177	1.520804	35.248247	75.125311	0.084151	0.661057	0.131370	95.446470	0.021973	0.550257
std	292.274324	1.127890	8.709026	13.263164	0.277680	0.473461	0.337883	5.900985	0.146629	0.497584
min	14.000000	0.000000	12.000000	31.000000	0.000000	0.000000	0.000000	70.000000	0.000000	0.000000
25%	727.000000	1.000000	29.000000	66.679200	0.000000	0.000000	0.000000	90.000000	0.000000	0.000000
50%	997.000000	2.000000	34.000000	74.390400	0.000000	1.000000	0.000000	100.000000	0.000000	1.000000
75%	1091.000000	3.000000	40.000000	82.555200	0.000000	1.000000	0.000000	100.000000	0.000000	1.000000
max	1231.000000	3.000000	70.000000	159.939360	1.000000	1.000000	1.000000	100.000000	1.000000	1.000000


8 rows × 11 columns

str2	strat	symptom	treat	offtrt	cd40	cd420	cd80	cd820	infected
2139.000000	2139.000000	2139.000000	2139.000000	2139.000000	2139.000000	2139.000000	2139.000000	2139.000000	2139.000000
0.585788	1.979897	0.172978	0.751286	0.362786	350.501169	371.307153	986.627396	935.369799	0.243572
0.492701	0.899053	0.378317	0.432369	0.480916	118.573863	144.634909	480.197750	444.976051	0.429338
0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	49.000000	40.000000	124.000000	0.000000
0.000000	1.000000	0.000000	1.000000	0.000000	263.500000	269.000000	654.000000	631.500000	0.000000
1.000000	2.000000	0.000000	1.000000	0.000000	340.000000	353.000000	893.000000	865.000000	0.000000
1.000000	3.000000	0.000000	1.000000	1.000000	423.000000	460.000000	1207.000000	1146.500000	0.000000
1.000000	3.000000	1.000000	1.000000	1.000000	1199.000000	1119.000000	5011.000000	6035.000000	1.000000

2.4. Kiểm tra dữ liệu trùng lặp

 `df.duplicated().sum()`

 0

- Kết luận không có dữ liệu trùng lặp.

CHƯƠNG 3: XỬ LÝ CÁC THUỘC TÍNH

3.1. Thuộc tính định lượng

3.1.1. time:

```
df['time'].describe()
```

```
count    2139.000000
mean      879.098177
std       292.274324
min       14.000000
25%       727.000000
50%       997.000000
75%      1091.000000
max      1231.000000
Name: time, dtype: float64
```

```
[ ] Q1 = np.quantile(df['time'], 0.25)
    Q1
```

```
727.0
```

```
[ ] Q3 = np.quantile(df['time'], 0.75)
    Q3
```

```
1091.0
```

```
[ ] IQR = Q3_age - Q1_age
    IQR
```

```
364.0
```

```
Range = df['time'].max() - df['time'].min()
Range
```

```
1217
```

```
[ ] df['time'].var()
```

```
85424.28034746922
```

```
[ ] df['time'].skew()
```

```
-1.1217076490553664
```

```
[ ] df['time'].kurtosis()
```

```
0.02623848883863067
```

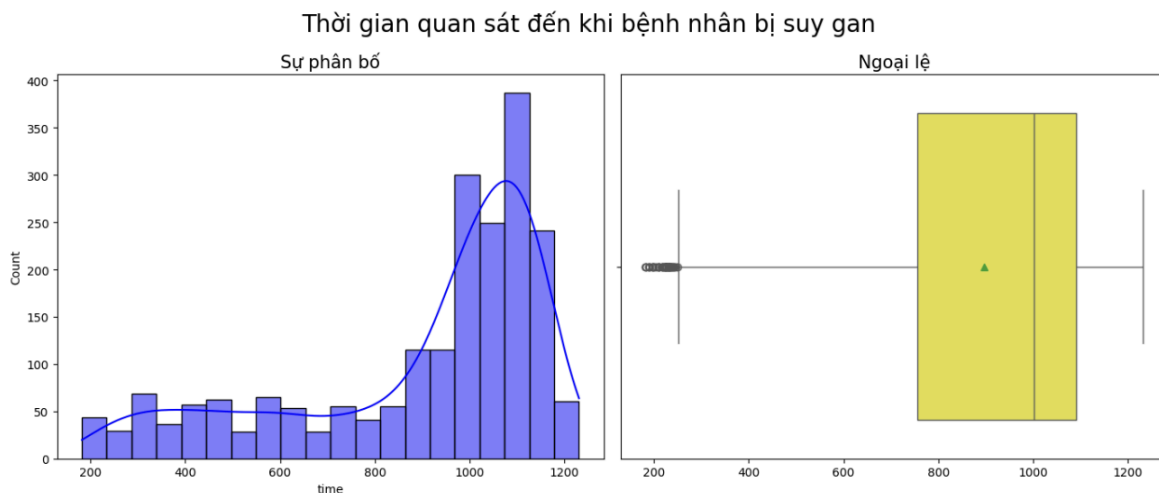
```

fig, axes = plt.subplots(1,2,figsize=(14, 6))
fig.suptitle('Thời gian quan sát đến khi bệnh nhân bị suy gan', fontsize=20)
axes[0] = sns.histplot(df, x = 'time', kde=True, color= 'blue', ax=axes[0])
axes[1] = sns.boxplot(x =df['time'], color='#FAF92F',showmeans=True, ax=axes[1])
axes[1].set(xlabel=None)

fontsize = 15
axes[0].set_title("Sự phân bố",fontdict={'fontsize': fontsize})
axes[1].set_title("Ngoại lệ",fontdict={'fontsize': fontsize})

plt.tight_layout()
plt.show()

```



- Nhận xét:

- Ta nhận thấy rằng có sự phân phối đa điểm phức tạp trong phân bố của dữ liệu. Điều này cho thấy, dữ liệu trên không phải là một phân phối đơn giản, cụ thể là nó có thể chứa các nhóm dữ liệu riêng biệt hoặc các cụm dữ liệu riêng biệt.
- Các giá trị ngoại lệ có thời gian quan sát ngắn (dưới khoảng 200 ngày).
- Các giá trị ngoại lệ không có vấn đề bất thường (từ 14-1231, theo thống kê từ kaggle).

3.1.2. age:

```
[ ] df['age'].describe()
```

```
count    2139.000000
mean      35.248247
std        8.709026
min       12.000000
25%       29.000000
50%       34.000000
75%       40.000000
max       70.000000
Name: age, dtype: float64
```

```
[ ] Q1 = np.quantile(df['age'], 0.25)
Q1
```

```
29.0
```

```
Q3 = np.quantile(df['age'], 0.75)
Q3
```

```
40.0
```

```
[ ] IQR = Q3 - Q1
IQR
```

```
11.0
```

```
[ ] Range = df['age'].max() - df['age'].min()
Range
```

```
58
```

```
[ ] df['age'].var()
```

```
75.8471379446521
```

```
[ ] df['age'].skew()
```

```
0.6424715060657828
```

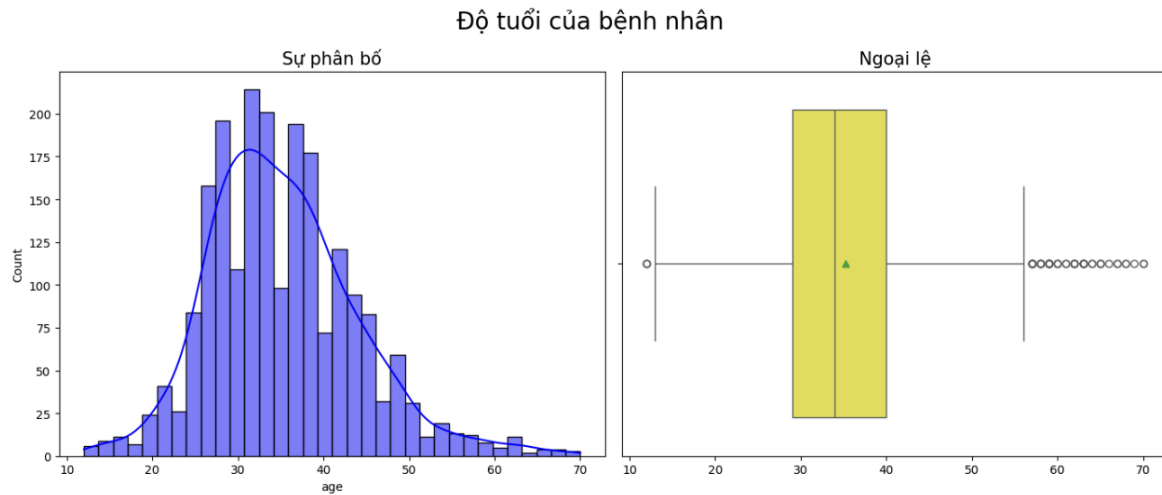
```
[ ] df['age'].kurtosis()
```

```
0.9776852790122885
```

```
fig, axes = plt.subplots(1,2,figsize=(14, 6))
fig.suptitle('Độ tuổi của bệnh nhân', fontsize=20)
axes[0]=sns.histplot(df, x='age', kde=True, color= 'blue', ax=axes[0])
axes[1]=sns.boxplot(x=df['age'], color='#FAF92F',showmeans=True, ax=axes[1])
axes[1].set(xlabel=None)

fontsize=15
axes[0].set_title("Sự phân bố",fontdict={'fontsize': fontsize})
```

```
axes[1].set_title("Ngoại lệ",fontdict={'fontsize': fontsize})
plt.tight_layout()
plt.show()
```



- Nhận xét:

- Ta nhận thấy đồ thị có xu hướng lệch phải.
- Bệnh nhân có độ tuổi nhiều nhất là từ 30 đến 40 tuổi.
- Các giá trị ngoại lệ có giá trị từ khoảng 55 đến 70 tuổi. Đây là các giá trị hợp lệ, nên ta sẽ không xử lý các ngoại lệ này.

3.1.3. wtkg:

```
[ ] df['wtkg'].describe()
```

```
count    2139.000000
mean      75.125311
std       13.263164
min       31.000000
25%       66.679200
50%       74.390400
75%       82.555200
max       159.939360
Name: wtkg, dtype: float64
```

```
[ ] Q1= np.quantile(df['wtkg'], 0.25)
Q1
```

```
66.6792
```

```
[ ] Q3 = np.quantile(df['wtkg'], 0.75)
Q3
```

```
⇒ 82.5552
```

```
[ ] IQR = Q3 - Q1
IQR
```

```
⇒ 15.876000000000005
```

```
[ ] Range = df['wtkg'].max() - df['wtkg'].min()
Range
```

```
⇒ 128.93936
```

```
[ ] df['wtkg'].var()
```

```
⇒ 175.91151938422513
```

```
[ ] df['wtkg'].skew()
```

```
⇒ 0.7064860599637041
```

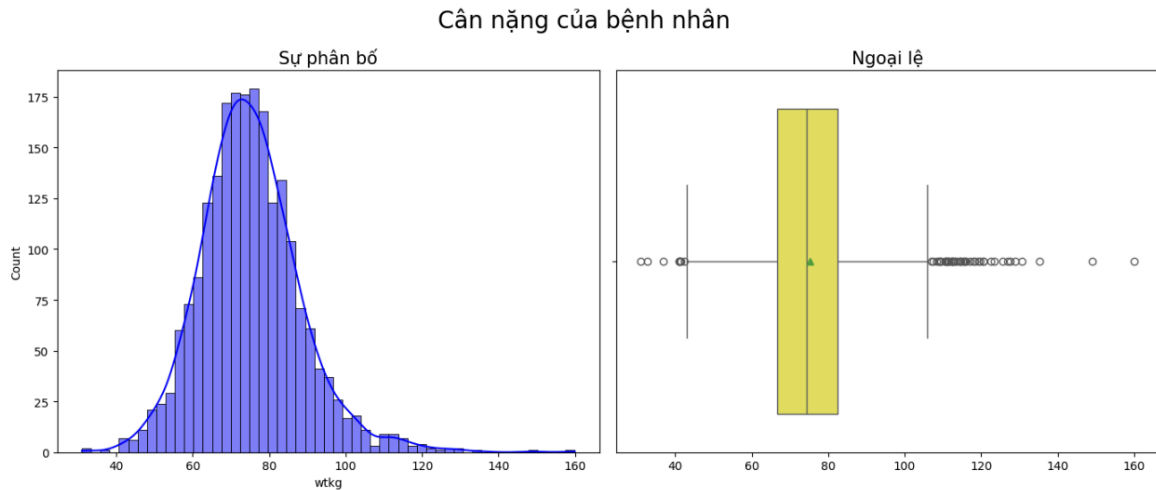
```
[ ] df['wtkg'].kurtosis()
```

```
⇒ 2.2290396224135995
```

```
fig, axes = plt.subplots(1,2,figsize=(14, 6))
fig.suptitle('Cân nặng của bệnh nhân', fontsize=20)
axes[0]=sns.histplot(df, x='wtkg', kde=True, color= 'blue', ax=axes[0])
axes[1]=sns.boxplot(x=df['wtkg'], color='#FAF92F',showmeans=True, ax=axes[1])
axes[1].set(xlabel=None)

fontsize=15
axes[0].set_title("Sự phân bố",fontdict={'fontsize': fontsize})
axes[1].set_title("Ngoại lệ",fontdict={'fontsize': fontsize})

plt.tight_layout()
plt.show()
```

- Nhận xét:

- Đồ thị có xu hướng lệch trái. Có phân phối chuẩn.
- Bệnh nhân có cân nặng chủ yếu từ 70-80kg.
- Có nhiều giá trị ngoại lệ, các giá trị ngoại lệ này nằm từ dưới 40 kg và trên 110 kg. Tuy nhiên, các giá trị cân nặng này vẫn tồn tại trong thực tế. Nên trong trường hợp này, ta sẽ không loại bỏ chúng.

3.1.4. karnofsky:

```
df['karnof'].describe()
```

```
count    2139.000000
mean      95.446470
std        5.900985
min        70.000000
25%        90.000000
50%       100.000000
75%       100.000000
max       100.000000
Name: karnof, dtype: float64
```

```
[ ] Q1 = np.quantile(df['karnof'], 0.25)
    Q1
```

```
90.0
```

```
[ ] Q3 = np.quantile(df['karnof'], 0.75)
    Q3
```

```
100.0
```

```
[ ] IQR = Q3 - Q1
    IQR
```

```
10.0
```

```
Range = df['karnof'].max() - df['karnof'].min()
Range
```

```
30
```

```
[ ] df['karnof'].var()
```

```
34.82161873286477
```

```
[ ] df['karnof'].skew()
```

```
-1.0276213559749043
```

```
[ ] df['karnof'].kurtosis()
```

```
0.6950742645693944
```

```
fig, axes = plt.subplots(1,2,figsize=(14, 6))
fig.suptitle('Điểm Karofsky của bệnh nhân', fontsize=20)
axes[0]=sns.histplot(df, x='karnof', kde=True, color= 'blue', ax=axes[0])
axes[1]=sns.boxplot(x=df['karnof'], color='#FAF92F',showmeans=True, ax=axes[1])
axes[1].set(xlabel=None)
```

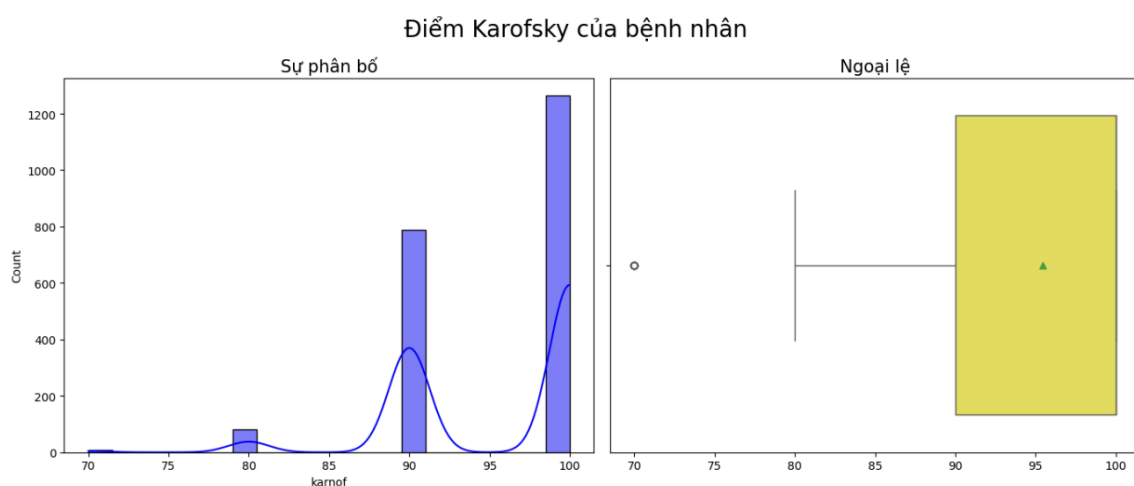
```
fontsize=15
```

```
axes[0].set_title("Sự phân bố",fontdict={'fontsize': fontsize})
```

```
axes[1].set_title("Ngoại lệ",fontdict={'fontsize': fontsize})
```

```
plt.tight_layout()
```

```
plt.show()
```



- Nhận xét:

- Phần lớn bệnh nhân có điểm số nằm trong khoảng 90 và khoảng 100.

- Chỉ có ngoại lệ duy nhất với các bệnh nhân có điểm số là 70. Tuy nhiên, điểm số này không phải là không thể xảy ra nên ta sẽ không loại bỏ nó.

3.1.5. preanti

```
[ ] df['preanti'].describe()
```

```
count    2139.000000
mean      379.175783
std       468.657526
min        0.000000
25%        0.000000
50%       142.000000
75%       739.500000
max      2851.000000
Name: preanti, dtype: float64
```

```
[ ] Q1 = np.quantile(df['preanti'], 0.25)
Q1
```

```
0.0
```

```
[ ] Q3 = np.quantile(df['preanti'], 0.75)
Q3
```

```
739.5
```

```
[ ] IQR = Q3 - Q1
IQR
```

```
739.5
```

```
Range = df['preanti'].max() - df['preanti'].min()
Range
```

```
2851
```

```
[ ] df['preanti'].var()
```

```
219639.8764759417
```

```
[ ] df['preanti'].skew()
```

```
1.1913747351308859
```

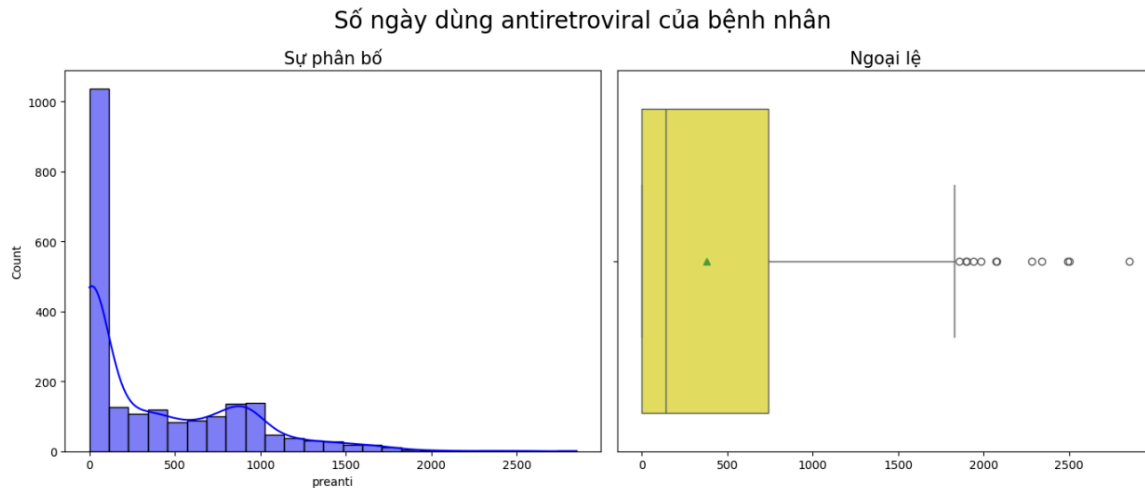
```
[ ] df['preanti'].kurtosis()
```

```
0.9392902282867901
```

```
fig, axes = plt.subplots(1,2,figsize=(14, 6))
fig.suptitle('Số ngày dùng antiretroviral của bệnh nhân', fontsize=20)
axes[0]=sns.histplot(df, x='preanti', kde=True, color= 'blue', ax=axes[0])
axes[1]=sns.boxplot(x=df['preanti'], color='#FAF92F',showmeans=True,
ax=axes[1])
axes[1].set(xlabel=None)

fontsize=15
axes[0].set_title("Sự phân bố",fontdict={'fontsize': fontsize})
axes[1].set_title("Ngoại lệ",fontdict={'fontsize': fontsize})
```

```
plt.tight_layout()
plt.show()
```



- Nhận xét:

- Hầu hết bệnh nhân có số ngày dùng antiretroviral là 0 ngày. Có thể là mới điều trị hay là không điều trị.
- Các giá trị ngoại lệ ở mức cao là trên 1250 ngày. Cần biết, liệu pháp antiretroviral là một phương pháp điều trị suốt đời cho cá bệnh nhân nhiễm HIV/AIDS, nên số ngày được điều trị có thể rất dài tùy thuộc vào tuổi thọ của bệnh nhân, nên những giá trị này sẽ không được xem là những giá trị vô lý và được giữ lại.

3.1.6. cd40:

```
df['cd40'].describe()
```

```
count    2139.000000
mean      350.501169
std       118.573863
min        0.000000
25%       263.500000
50%       340.000000
75%       423.000000
max      1199.000000
Name: cd40, dtype: float64
```

```
[ ] Q1 = np.quantile(df['cd40'], 0.25)
Q1
```

```
263.5
```

```
[ ] Q3 = np.quantile(df['cd40'], 0.75)
Q3
```

```
423.0
```

```
[ ] IQR = Q3 - Q1
IQR
```

```
159.5
```

```
[ ] Range = df['cd40'].max() - df['cd40'].min()
Range
```

```
1199
```

```
[ ] df['cd40'].var()
```

```
14059.760873282543
```

```
df['cd40'].skew()
```

```
0.7578678172259868
```

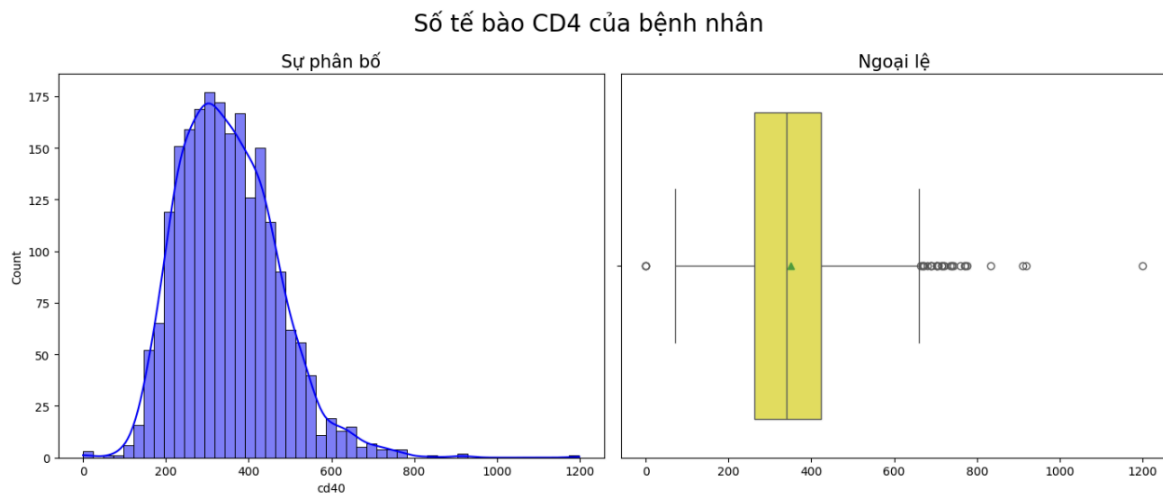
```
[ ] df['cd40'].kurtosis()
```

```
1.805410671463028
```

```
fig, axes = plt.subplots(1,2,figsize=(14, 6))
fig.suptitle('Số tế bào CD4 của bệnh nhân', fontsize=20)
axes[0]=sns.histplot(df, x='cd40', kde=True, color= 'blue', ax=axes[0])
axes[1]=sns.boxplot(x=df['cd40'], color='#FAF92F',showmeans=True, ax=axes[1])
axes[1].set(xlabel=None)

fontsize=15
axes[0].set_title("Sự phân bố",fontdict={'fontsize': fontsize})
axes[1].set_title("Ngoại lệ",fontdict={'fontsize': fontsize})

plt.tight_layout()
plt.show()
```



- Nhận xét:

- Có sự phân bố đa điểm trong đồ thị phân bố của số tế bào CD4 của bệnh nhân.
- Ta có các giá trị ngoại lệ ở mức thấp và mức cao (hơn 650 tế bào).
- Theo nghiên cứu, đối với người khỏe mạnh thì tế bào có thể mức rất cao (trên 1000). Vì thế những giá trị ngoại lệ ở mức cao sẽ được giữ lại.
- Với những giá trị ngoại lệ ở mức thấp, ta sẽ kiểm tra min của dữ liệu. Cần biết giá trị tế bào cd4 có thể gần với mức 0 do nhiễm AIDS. Tuy nhiên, việc chỉ số tế bào này có giá trị 0 là gần như không thể, thế nên ta sẽ loại bỏ những record có giá trị 0 này.

- Loại bỏ các giá trị 0:

```
[ ] # Loại bỏ các giá trị 0
df = df[(df['cd40'] > 0)]
df.shape
```

(2136, 23)

3.1.7. cd420

```
[ ] df['cd420'].describe()
```

```
count    2136.000000
mean      371.150749
std       144.637561
min        49.000000
25%       269.000000
50%       353.000000
75%       460.000000
max      1119.000000
Name: cd420, dtype: float64
```

```
[ ] Q1 = np.quantile(df['cd420'], 0.25)
Q1
```

```
269.0
```

```
[ ] Q3 = np.quantile(df['cd420'], 0.75)
Q3
```

```
460.0
```

```
[ ] IQR = Q3 - Q1
IQR
```

```
191.0
```

```
[ ] Range = df['cd420'].max() - df['cd420'].min()
Range
```

```
1070
```

```
[ ] df['cd420'].var()
```

```
20920.024102483138
```

```
df['cd420'].skew()
```

```
0.7328966226757088
```

```
[ ] df['cd420'].kurtosis()
```

```
1.077551194951814
```

```
fig, axes = plt.subplots(1,2,figsize=(14, 6))
fig.suptitle('Số tế bào CD4 của bệnh nhân trong 20+/5 tuần', fontsize=20)
axes[0]=sns.histplot(df, x='cd420', kde=True, color= 'blue', ax=axes[0])
axes[1]=sns.boxplot(x=df['cd420'], color='#FAF92F',showmeans=True, ax=axes[1])
axes[1].set(xlabel=None)
```

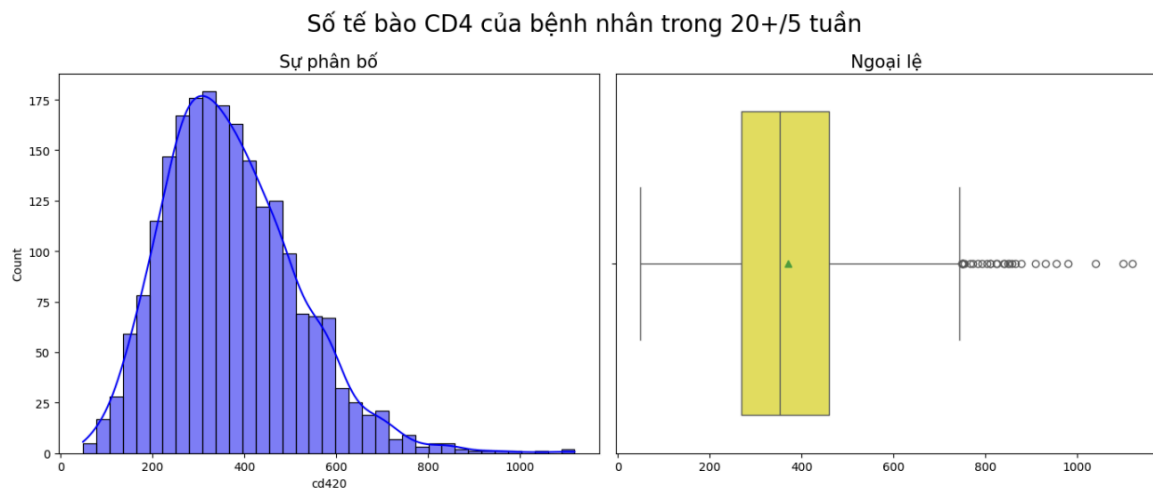
```
fontsize=15
```

```
axes[0].set_title("Sự phân bố",fontdict={'fontsize': fontsize})
```

```
axes[1].set_title("Ngoại lệ",fontdict={'fontsize': fontsize})
```

```
plt.tight_layout()
```

```
plt.show()
```



- Nhận xét:
 - Đồ thị có sự lệch phải.
 - Các giá trị ngoại lệ nằm ở mức cao (khoảng hơn 850).
 - Như đã giải thích ở trên, chúng ta sẽ không loại các giá trị ngoại lệ này.

3.1.8. cd80:

```
df['cd80'].describe()
```

count	2136.000000
mean	986.817884
std	480.398891
min	40.000000
25%	654.000000
50%	893.000000
75%	1208.000000
max	5011.000000
Name: cd80, dtype: float64	

```
[ ] Q1 = np.quantile(df['cd80'], 0.25)
Q1
```

```
654.0
```

```
[ ] Q3 = np.quantile(df['cd80'], 0.75)
Q3
```

```
1208.0
```



```
[ ] IQR = Q3 - Q1  
IQR
```

```
↔ 554.0
```

```
[ ] Range = df['cd80'].max() - df['cd80'].min()  
Range
```

```
↔ 4971
```

```
[ ] df['cd80'].var()
```

```
↔ 230783.09421865817
```

```
[ ] df['cd80'].skew()
```

```
↔ 1.7333893732347885
```

```
[ ] df['cd80'].kurtosis()
```

```
↔ 6.173129079862598
```

```
fig, axes = plt.subplots(1,2,figsize=(14, 6))  
fig.suptitle('Số tế bào CD8 của bệnh nhân', fontsize=20)  
axes[0]=sns.histplot(df, x='cd80', kde=True, color= 'blue', ax=axes[0])  
axes[1]=sns.boxplot(x=df['cd80'], color='#FAF92F',showmeans=True, ax=axes[1])  
axes[1].set(xlabel=None)
```

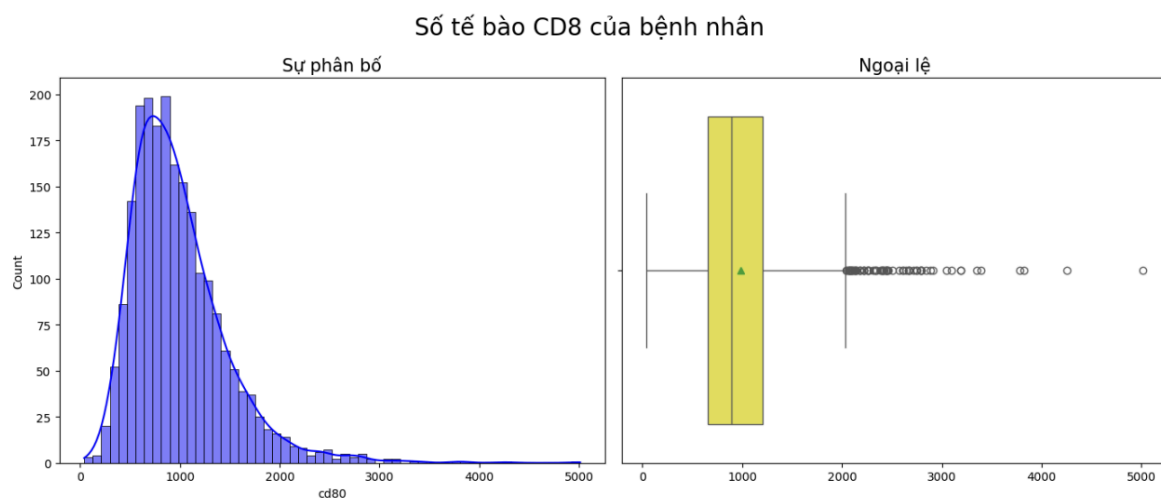
```
fontsize=15
```

```
axes[0].set_title("Sự phân bố",fontdict={'fontsize': fontsize})
```

```
axes[1].set_title("Ngoại lệ",fontdict={'fontsize': fontsize})
```

```
plt.tight_layout()
```

```
plt.show()
```



- Nhận xét:
 - Đồ thị có xu hướng lệch phải.
 - Giá trị ngoại lệ ở mức cao (trên 2000 tế bào).
 - Tương tự như cd4, các tế bào cd8 cũng có thể ở mức cao biểu thị cho cơ thể khỏe mạnh. Nên những giá trị này không được xem là ngoại lệ.

3.1.9. cd820:

```
df['cd820'].describe()
```

```
count    2136.000000
mean      935.061798
std       444.907840
min       124.000000
25%       630.750000
50%       865.000000
75%      1146.250000
max       6035.000000
Name: cd820, dtype: float64
```

```
[ ] Q1 = np.quantile(df['cd820'], 0.25)
Q1
```

```
630.75
```

```
[ ] Q3 = np.quantile(df['cd820'], 0.75)
Q3
```

```
1146.25
```

```
[ ] IQR = Q3 - Q1
IQR
```

```
515.5
```

```
[ ] Range = df['cd820'].max() - df['cd820'].min()
Range
```

```
5911
```

```
[ ] df['cd820'].var()
```

```
197942.9863431834
```

```
df['cd820'].skew()
```

```
2.09486626222449
```

```
[ ] df['cd820'].kurtosis()
```

```
11.801725505568552
```

```
fig, axes = plt.subplots(1,2,figsize=(14, 6))
fig.suptitle('Số tế bào CD8 của bệnh nhân trong 20+/5 tuần', fontsize=20)
axes[0]=sns.histplot(df, x='cd820', kde=True, color= 'blue', ax=axes[0])
axes[1]=sns.boxplot(x=df['cd820'], color='#FAF92F',showmeans=True, ax=axes[1])
```

```
axes[1].set(xlabel=None)
```

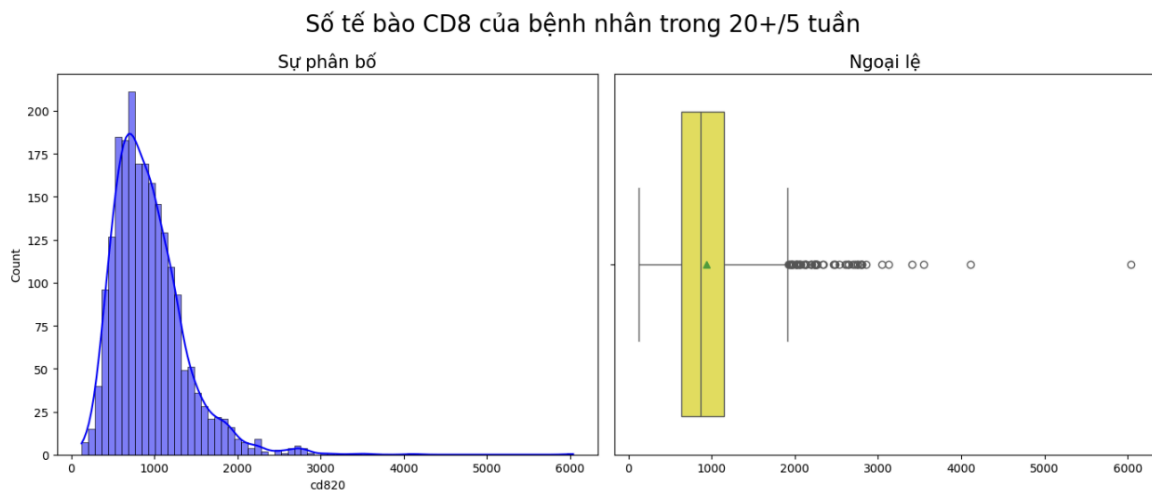
```
fontsize=15
```

```
axes[0].set_title("Sự phân bố",fontdict={'fontsize': fontsize})
```

```
axes[1].set_title("Ngoại lệ",fontdict={'fontsize': fontsize})
```

```
plt.tight_layout()
```

```
plt.show()
```



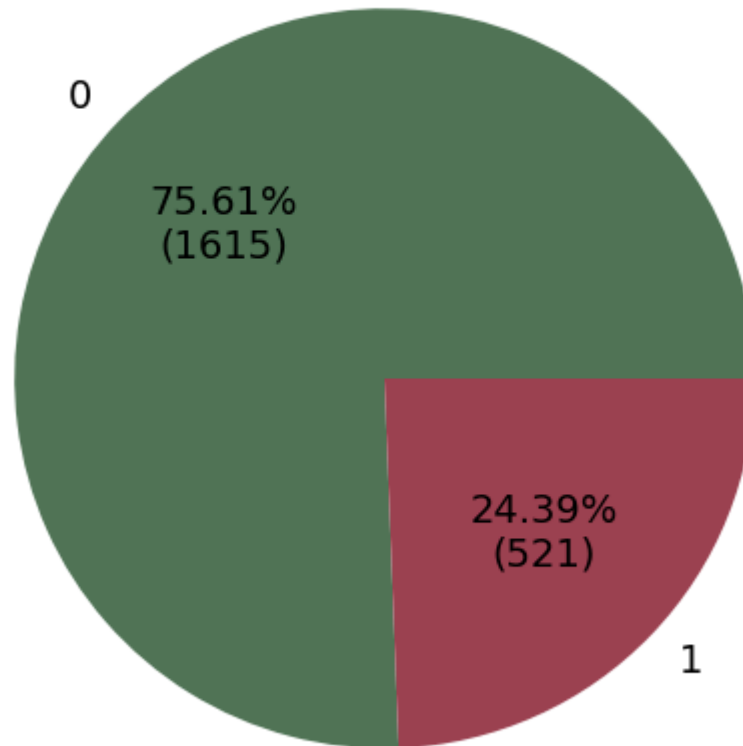
- Nhận xét:

- Đồ thị có sự lệch phải.
- Các giá trị ngoại lệ nằm ở mức cao (khoảng hơn 1750).
- Tương tự, ta cũng sẽ không loại những giá trị này.

3.2. Thuộc tính phân loại

3.2.1. infected:

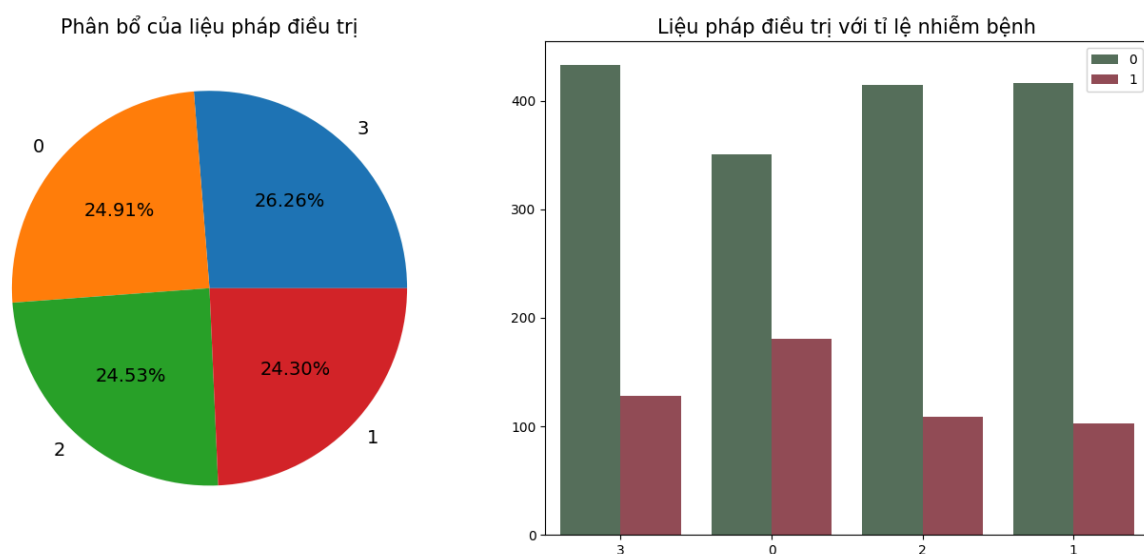
Phân bố của tỉ lệ nhiễm bệnh



Nhận xét:

- Số người không nhiễm có tỉ lệ cao hơn số người bị nhiễm.

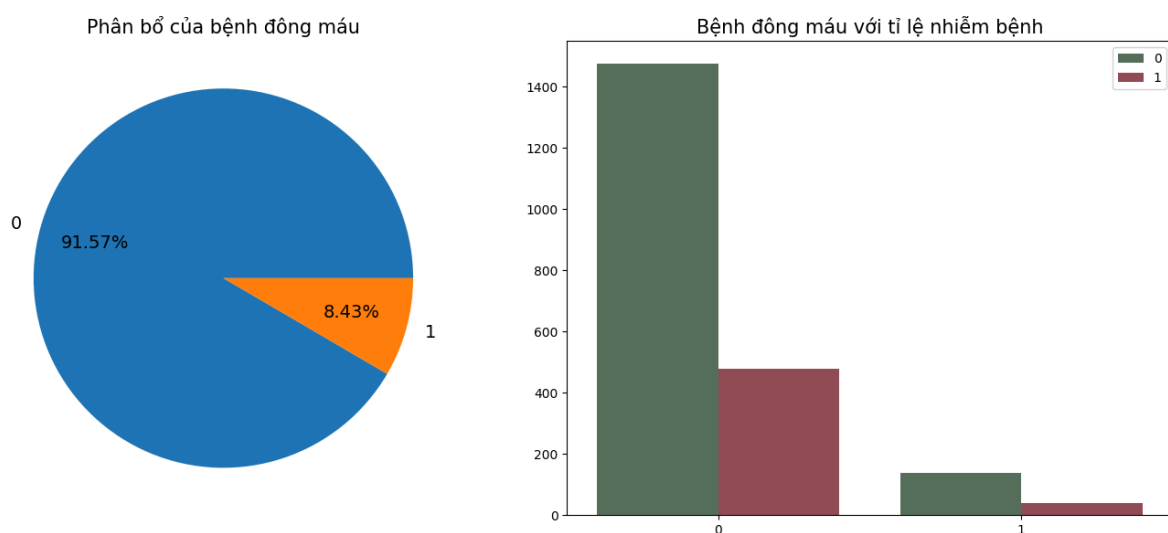
3.2.2. trt:



Nhận xét:

- Số người chỉ sử dụng ddl là cao nhất.
- Những người chỉ sử dụng ZDV có số lượng nhiễm bệnh cao hơn những liệu pháp điều trị khác. Điều này có thể là do những phương pháp điều trị khác có thể cho ra hiệu quả cao hơn so với việc chỉ dùng mỗi ZDV.

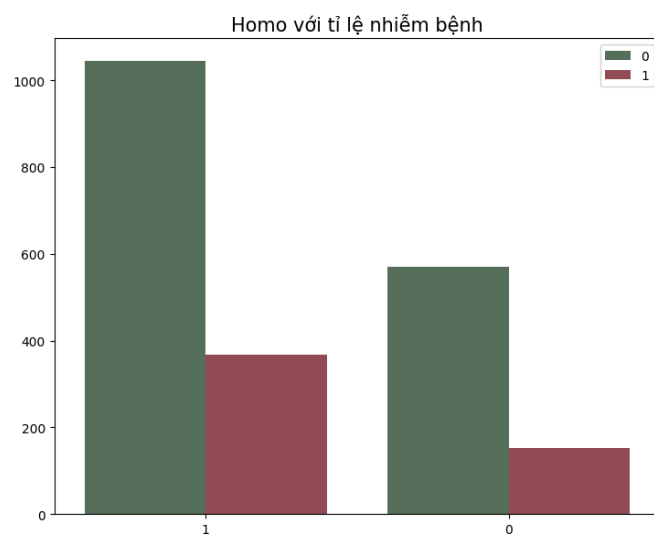
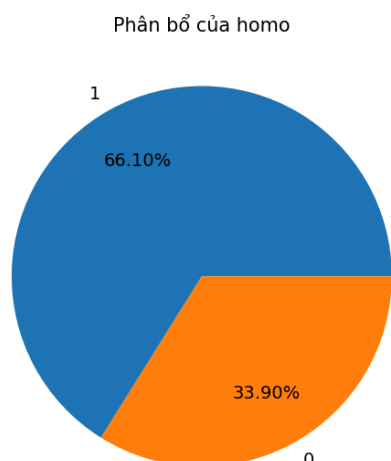
3.2.3. hemo:



Nhận xét:

- Số người không có triệu chứng đông máu chiếm đáng kể.
- Số người có triệu chứng chỉ chiếm phần ít (khoảng 8.43%).

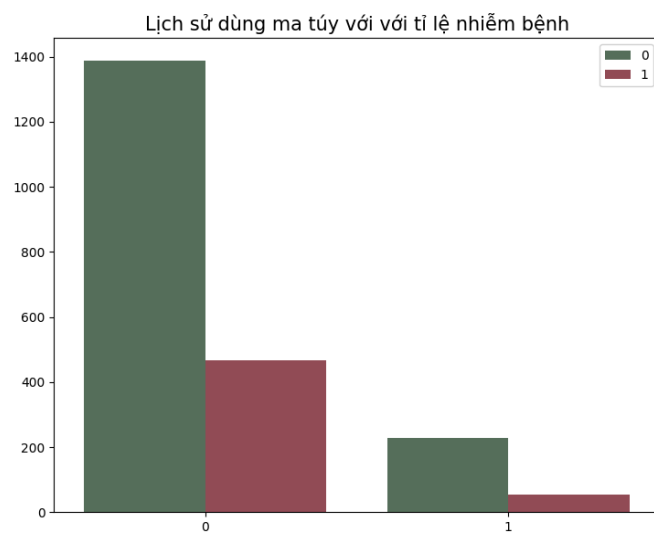
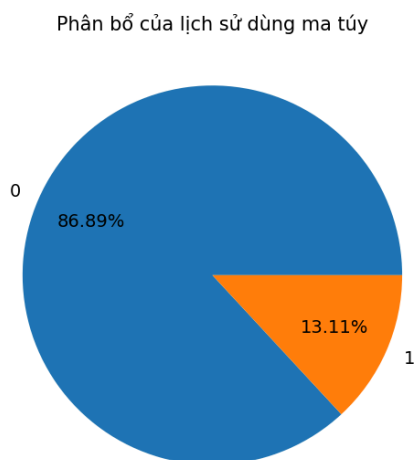
3.2.4.homo:



Nhận xét:

- Người có hoạt động đồng tính chiếm tỉ lệ cao hơn.

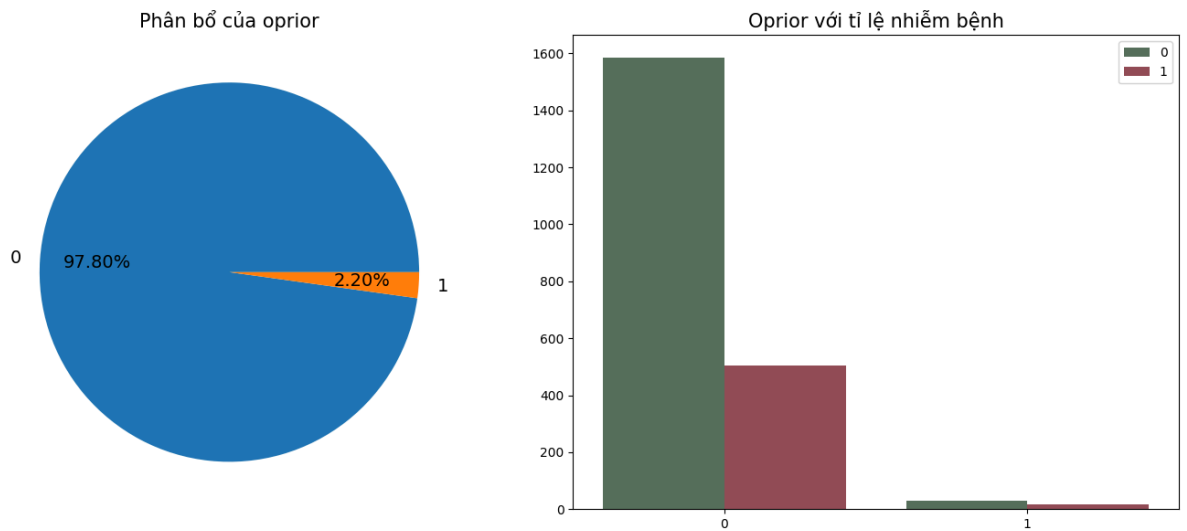
3.2.5. drugs:



Nhận xét:

- Số người không dùng ma túy chiếm nhiều hơn.

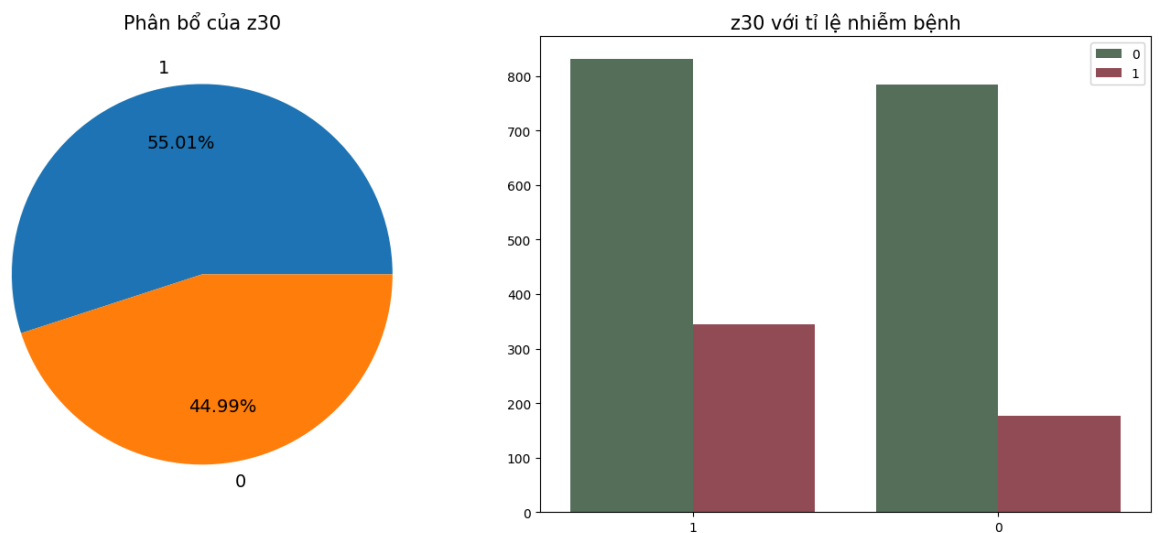
3.2.6. oprior:



Nhận xét:

- Số người không dùng nhiều hơn.

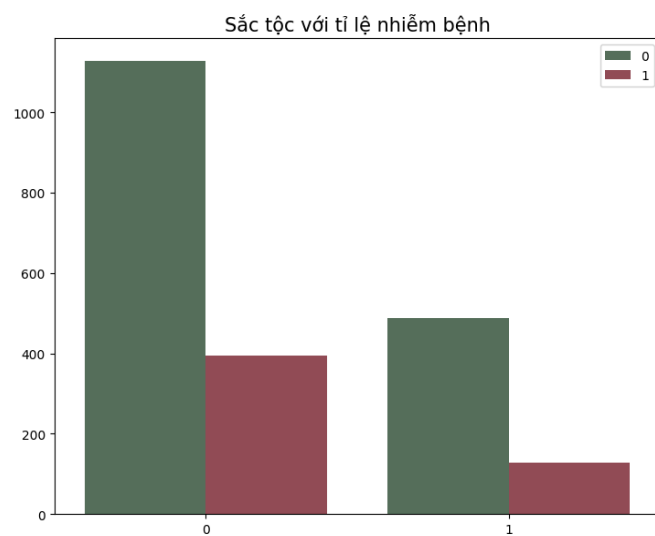
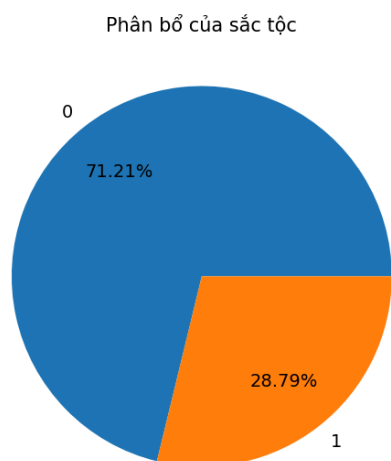
3.2.7. z30:



Nhận xét:

- Số người dùng nhiều hơn.
- Ta nhận thấy, số người chỉ dùng ZDV và dùng các phương pháp khác không có sự chênh lệch nhau quá nhiều, tuy nhiên ở phía những người chỉ dùng ZDV, số người nhiễm lại cao gấp đôi. Điều này một lần nữa lại cho thấy các phương pháp khác mang lại hiệu quả cao hơn so với chỉ dùng ZDV.

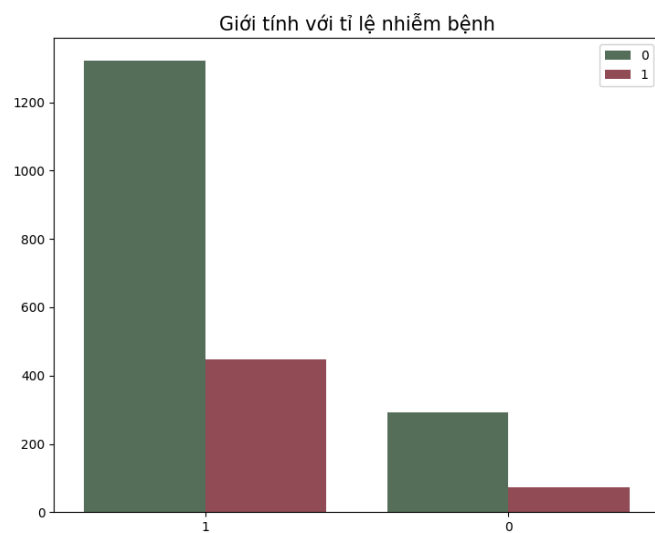
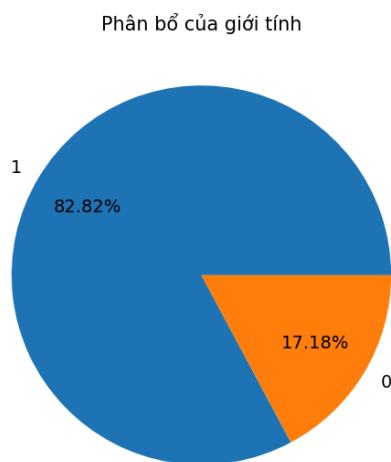
3.2.8. race:



Nhận xét:

- Bệnh nhân có sắc tộc da trắng chiếm tỉ lệ cao hơn.

3.2.9. gender:

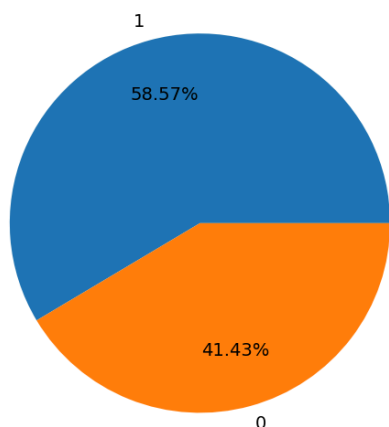


Nhận xét:

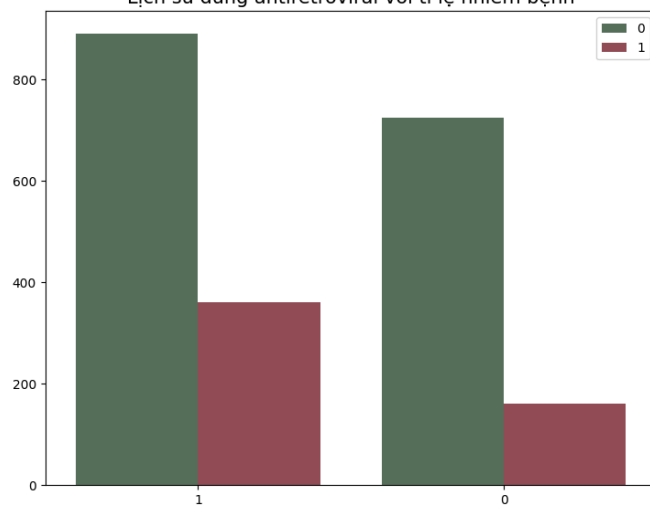
- Bệnh nhân nữ chiếm tỉ lệ cao hơn.

3.2.10. str2:

Phân bố của lịch sử dùng antiretroviral



Lịch sử dùng antiretroviral với tỉ lệ nhiễm bệnh

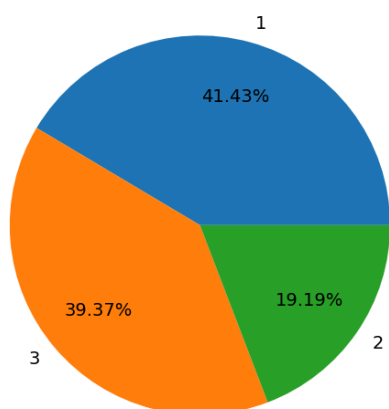


Nhận xét:

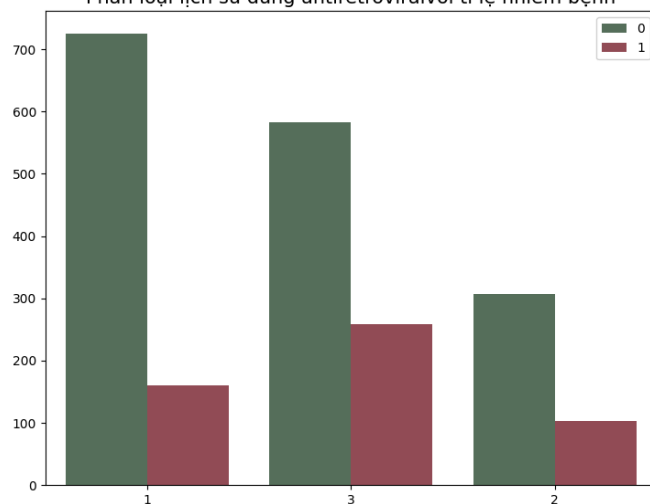
- Những người sử dụng cao hơn.
- Những người đã trải nghiệm dùng antiretroviral có số người nhiễm bệnh cao hơn. Do liệu pháp này chủ yếu dành cho người nghi nhiễm HIV/AIDS nên những người đã trải nghiệm qua thì tỉ lệ họ đang nhiễm cũng sẽ cao hơn.

3.2.11. strat:

Phân bố của phân loại lịch sử dùng antiretroviral



Phân loại lịch sử dùng antiretroviral với tỉ lệ nhiễm bệnh



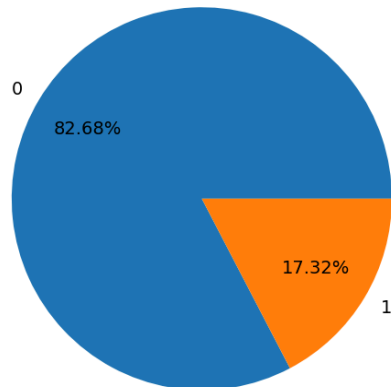
Nhận xét:

- Đa số là những người chưa dùng liệu pháp hoặc dùng liệu pháp trên 52 tuần.

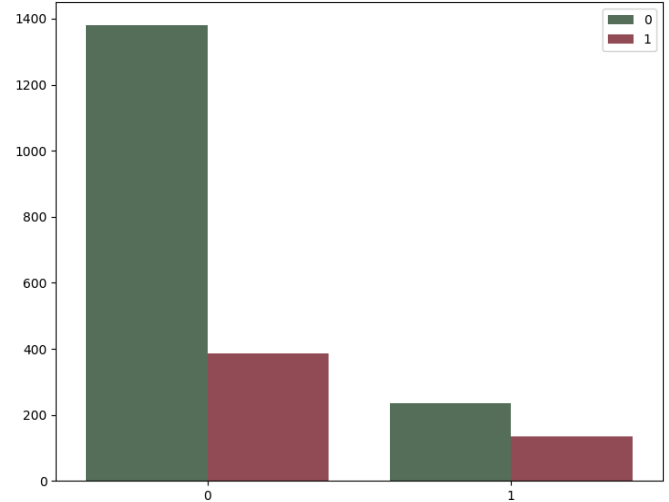
- Những người chưa dùng liệu pháp mặc dù chiếm tỉ lệ cao nhất nhưng lại có số người nhiễm thấp hơn.
- Ngược lại, những người sử dụng liệu pháp này có số người nhiễm cao hơn (giống như phân tích ở phần str2 trước đó).

3.2.12. symptom:

Phân bố của những người có triệu chứng



Những người có triệu chứng với tỉ lệ nhiễm bệnh

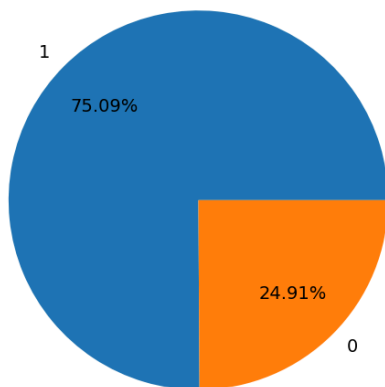


Nhận xét:

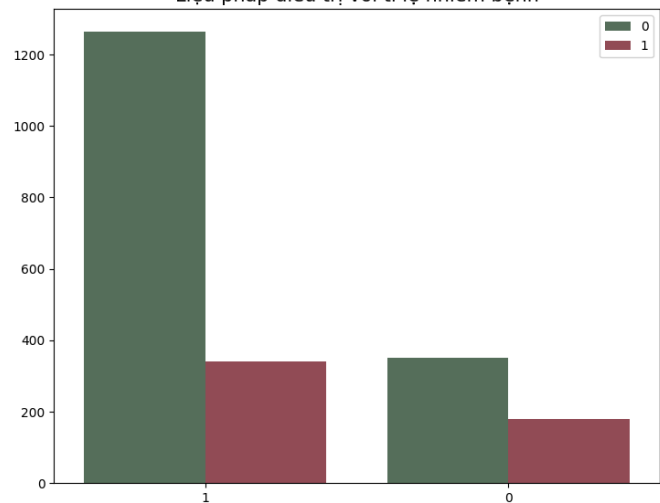
- Đa số là những người không có triệu chứng. Trong tổng số những người có triệu chứng, tỉ lệ người nhiễm cao hơn (50%).

3.2.13. treat:

Phân bố của liệu pháp điều trị



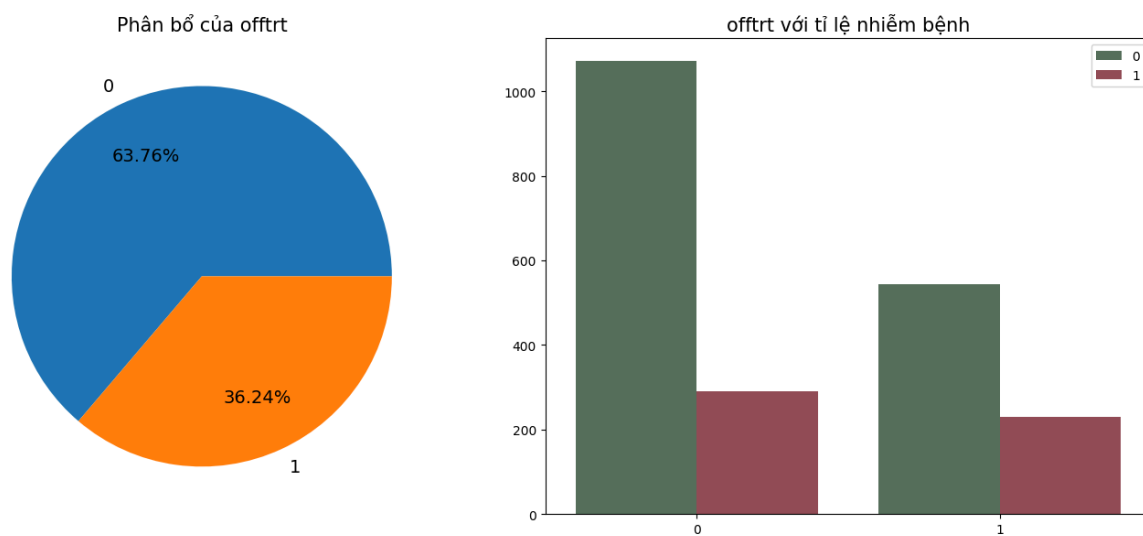
Liệu pháp điều trị với tỉ lệ nhiễm bệnh



Nhận xét:

- Hầu hết bệnh nhân sử dụng các liệu pháp khác so với việc chỉ dùng mỗi ZDV.
- Trong tổng số những người chỉ sử dụng ZDV, tỉ lệ người nhiễm chiếm 50%.

3.2.14. offtrt:

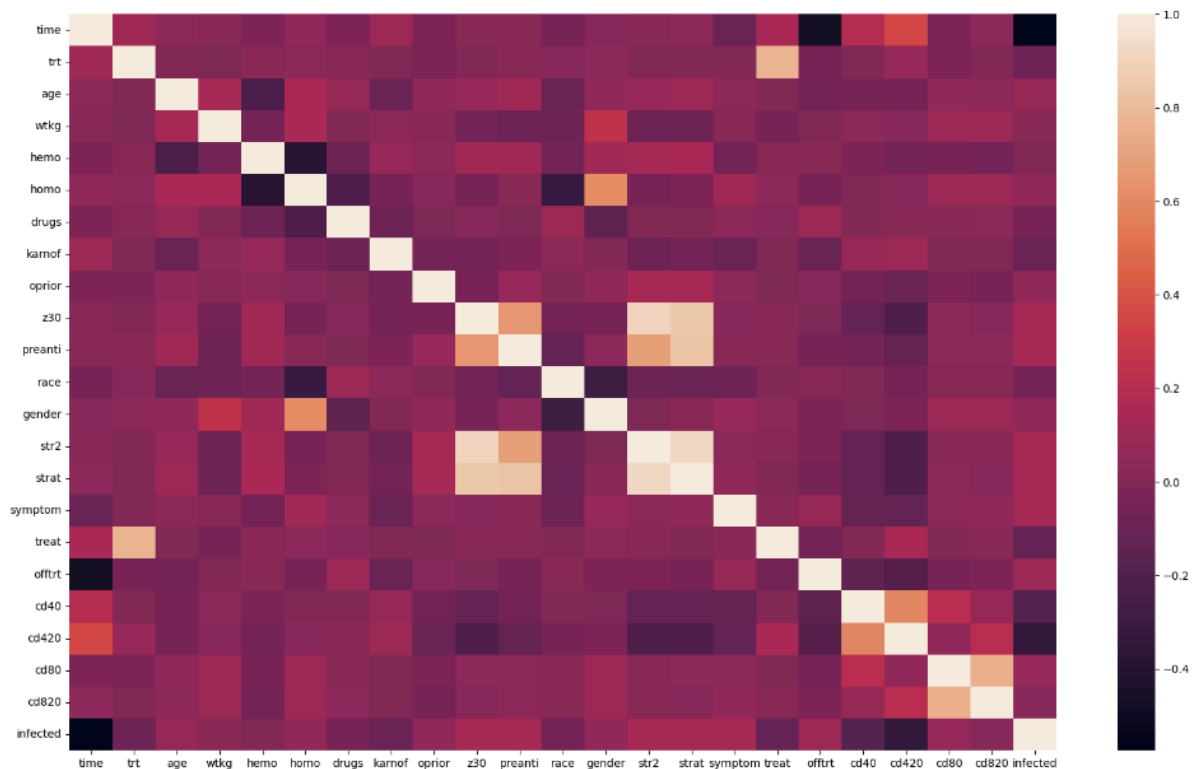


Nhận xét:

- Đã số là những người không ngừng việc chữa trị trước thời điểm 96 +/- 5 tuần.
- Những người ngừng trước thời điểm này có tỉ lệ người nhiễm cao hơn (gần 50%). Điều này cho thấy, liệu pháp antiretroviral này đóng góp một phần quan trọng trong việc ngăn ngừa virus phát tán.

3.3. Biểu đồ tương quan

```
plt.figure(figsize=(20, 12))
sns.heatmap(df.corr(), annot=False)
plt.show()
```



CHƯƠNG 4: HUẤN LUYỆN MÔ HÌNH DỰ ĐOÁN KHẢ NĂNG MẮC BỆNH AIDS

4.1. Chuẩn bị tập dữ liệu

- Chia tập dữ liệu thành tập đặc trưng (X) và tập mục tiêu đầu ra (Y).

```
[ ] df.shape
(2136, 23)

[ ] # Chia tập dữ liệu thành tập đặc trưng (X) và tập mục tiêu đầu ra (Y)
x = df.drop('infected', axis=1)
y = df['infected']
```

- Chia tập huấn luyện và tập kiểm tra.

```
[ ] # Chia tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

↗ ((1495, 22), (1495,)), (374, 22), (374,))

- Cân bằng dữ liệu.

```
[ ] # Cân bằng dữ liệu
smote = SMOTE(random_state=14)
X_train, y_train = smote.fit_resample(X_train, y_train)
```

▶ X_train.shape, y_train.shape, X_test.shape, y_test.shape

↗ ((2278, 22), (2278,)), (374, 22), (374,))

4.2. Scale dữ liệu

- Ở phần EDA trước đó, chúng ta thấy rằng các giá trị có rất nhiều sự khác biệt. Như tuổi thì nằm trong khoảng 40-60, time thì 600-1200. Thế nên việc scale dữ liệu là cần thiết để giúp cho quá trình train dữ liệu được tốt hơn.
- Cũng ở phần EDA, ta nhận thấy các dữ liệu hầu như không có phân phối chuẩn. Nên thích hợp nhất là ta sẽ dùng MinMaxScaler để scale dữ liệu.

```
▶ scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
X_train
```

↗ array([[0.78060805, 0.66666667, 0.44186047, ..., 0.49493488, 0.27792483,
0.43925793],
[0.9104355 , 1. , 0.34883721, ..., 0.34442836, 0.41926945,
0.78037104],
[0.60476582, 0.66666667, 0.3255814 , ..., 0.34587554, 0.31392271,
0.16337522],
...,
[0.24322104, 0. , 0.39534884, ..., 0.24457308, 0.53043939,
0.73608618],
[0.78389482, 1. , 0.53488372, ..., 0.37047757, 0.28798306,
0.16098145],
[0.56943303, 0.33333333, 0.74418605, ..., 0.45151954, 0.80518793,
0.50628366]])

4.3. Xây dựng mô hình

Ở phần xây dựng mô hình này, ta sẽ dùng AutoLogging-ML để chạy các kết quả dự đoán với các mô hình khác nhau, sau đó ta sẽ lựa ra 3 mô hình có điểm số cao nhất, xây dựng lại các mô hình đó để cải thiện các đánh giá của mô hình.

```
!pip install AutoLogging-ML
!pip install catboost

Requirement already satisfied: AutoLogging-ML in /usr/local/lib/python3.10/dist-packages (1.2.4)
Requirement already satisfied: catboost in /usr/local/lib/python3.10/dist-packages (1.2.5)
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (from catboost) (0.20.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from catboost) (3.7.1)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.25.2)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist-packages (from catboost) (2.0.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from catboost) (1.11.4)
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (from catboost) (5.15.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from catboost) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (4.51.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (24.0)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (3.1.2)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly->catboost) (8.3.0)
```

```
from AutoLogging_ML import AutoLogger
r,model =AutoLogger.train_and_log_classification(X_train,y_train,X_test,y_test,
size='large')
```

	model	training- accuracy	training- precision	training- recall	training- f1	training- confusion matrix	validation- accuracy	validation- precision	validation- recall	validation- f1	validation- confusion matrix	training- classification report	validation- classification report
0	naive bayes	81.518876	81.695393	81.518876	81.493109	[[886, 253], [168, 971]]	72.459893	67.211082	71.360182	67.896996	[[206, 74], [29, 65]]	precision recall f1-score ...	precision recall f1-score ...
10	quadratic discriminant analysis	86.040386	86.051502	86.040386	86.039310	[[970, 166], [149, 990]]	75.401070	69.060630	72.264438	70.057087	[[220, 60], [32, 62]]	precision recall f1-score ...	precision recall f1-score ...
1	decision tree	100.000000	100.000000	100.000000	100.000000	[[1139, 0], [0, 1139]]	83.957219	78.572993	79.745441	79.120920	[[247, 33], [27, 67]]	precision recall f1-score ...	precision recall f1-score ...
6	logistic regression	87.357331	87.390649	87.357331	87.354514	[[1012, 127], [161, 978]]	85.561497	80.492424	83.643617	81.801471	[[245, 35], [19, 75]]	precision recall f1-score ...	precision recall f1-score ...
7	bagging classifier	99.604917	99.604955	99.604917	99.604917	[[1135, 4], [5, 1134]]	86.096257	81.366460	82.234043	81.782074	[[252, 28], [24, 70]]	precision recall f1-score ...	precision recall f1-score ...
9	linear discriminant analysis	87.093942	87.268720	87.093942	87.078793	[[1031, 108], [186, 953]]	86.631016	81.942020	83.651216	82.715912	[[251, 29], [21, 73]]	precision recall f1-score ...	precision recall f1-score ...
3	adaboost	91.922739	91.969454	91.922739	91.920491	[[1028, 111], [73, 1066]]	87.165775	82.442103	87.895137	84.377611	[[242, 38], [10, 84]]	precision recall f1-score ...	precision recall f1-score ...
5	xgboost	100.000000	100.000000	100.000000	100.000000	[[1139, 0], [0, 1139]]	87.967914	84.079516	83.837386	83.957219	[[258, 22], [23, 71]]	precision recall f1-score ...	precision recall f1-score ...
8	extra trees classifier	100.000000	100.000000	100.000000	100.000000	[[1139, 0], [0, 1139]]	88.502674	85.501730	83.134498	84.210268	[[263, 17], [26, 68]]	precision recall f1-score ...	precision recall f1-score ...
2	random forest	100.000000	100.000000	100.000000	100.000000	[[1139, 0], [0, 1139]]	89.572193	85.683458	87.382219	86.473779	[[257, 23], [16, 78]]	precision recall f1-score ...	precision recall f1-score ...
4	gradient boost	95.478490	95.478805	95.478490	95.478482	[[1089, 50], [53, 1086]]	90.106952	85.936697	89.859422	87.568612	[[253, 27], [10, 84]]	precision recall f1-score ...	precision recall f1-score ...

Tiêu chí lựa chọn mô hình

- Đảm bảo việc dự đoán trên tập train cao hơn 90%.
- Do đây là mô hình dùng để dự đoán bệnh bên y tế, nên ta sẽ quan tâm các chỉ số về accuracy và recall. Đặc biệt là recall, cần độ nhạy cao để có thể phát hiện và điều trị sớm.

Kết luận: Dựa trên các tiêu chí đánh giá trên chúng ta có thể kết luận rằng 3 mô hình tốt nhất bao gồm Gradient boost, Random Forest và Adaboost.

4.4. Cải thiện mô hình

- Hàm `accuracy_score` từ thư viện `sklearn` được sử dụng để tính toán độ chính xác của mô hình dự đoán. Nó so sánh các giá trị trong `y_pred` (các dự đoán của mô hình) với các giá trị trong `y_test` (các nhãn đúng). Kết quả độ chính xác được gán vào biến `accuracy`.
- Hàm `classification_report` từ thư viện `sklearn` được sử dụng để tạo báo cáo chi tiết về kết quả phân loại của mô hình. Nó so sánh `y_pred` với `y_test` và in ra báo cáo này.
- Hàm `confusion_matrix` từ thư viện `sklearn` được sử dụng để tính ma trận nhầm lẫn (confusion matrix). Nó so sánh `y_pred` với `y_test` và trả về một ma trận biểu diễn số lượng các dự đoán đúng và sai.
- Dùng thư viện `seaborn` (được import với tên gọi `sns`) để vẽ một biểu đồ heatmap dựa trên ma trận nhầm lẫn. Heatmap này sẽ hiển thị các giá trị trong ma trận dưới dạng màu sắc. `annot=True` để hiển thị giá trị của từng ô trong biểu đồ. `fmt='d'` để định dạng giá trị hiển thị là số nguyên. `cmap='Blues'` để sử dụng màu xanh lam cho biểu đồ. `xticklabels` và `yticklabels` được sử dụng để đặt tên cho các trục x và y của biểu đồ.

```
def danh_gia(y_test, y_pred):
    accuracy = accuracy_score(y_test, y_pred)
    print(f'Độ chính xác: {accuracy}')
    print("Báo cáo phân loại:")
    print(classification_report(y_test, y_pred))

    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Predicted
Negative', 'Predicted Positive'], yticklabels=['Actual Negative', 'Actual Positive'])
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title('Confusion Matrix')
    plt.show()
```

- Sử dụng phương thức `predict_proba` của `model` để dự đoán xác suất của các lớp trong `X_test`. Ở đây, `[:, 1]` được sử dụng để lấy chỉ xác suất của lớp positive (1).
- Hàm `roc_curve` từ thư viện `sklearn` được sử dụng để tính toán các giá trị False Positive Rate (fpr), True Positive Rate (tpr) và ngưỡng (thresholds) tương ứng. Nó so sánh `y_prob` với `y_test` để tính toán các giá trị này.

```

def plot_roc(model, X_test, y_test):
    y_prob = model.predict_proba(X_test)[: , 1]
    fpr, tpr, thresholds = roc_curve(y_test, y_prob)

    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (area = %0.2f)' %
roc_auc_score(y_test, y_prob))
    plt.plot([0, 1], [0, 1], color='grey', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC)')
    plt.legend(loc="lower right")
    plt.show()

```

4.4.1. Random Forest Classifier

- Tạo một đối tượng RandomForestClassifier từ thư viện sklearn. Đối số n_estimators=100 chỉ định số lượng cây trong mô hình RandomForest và random_state=42 đặt seed cho việc tạo ngẫu nhiên cây trong mô hình để đảm bảo kết quả tái lập được.
- Sử dụng phương thức fit của model để huấn luyện mô hình RandomForest trên dữ liệu huấn luyện. X_train là ma trận dữ liệu đặc trưng huấn luyện và y_train là vector nhãn tương ứng.
- Sử dụng phương thức predict của model để dự đoán nhãn cho dữ liệu X_test. Kết quả dự đoán được gán vào biến y_pred.

```

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
y_pred = rf_model.predict(X_test)
danh_gia(y_test, y_pred)

```


Độ chính xác: 0.866822429906542				
Báo cáo phân loại:				
	precision	recall	f1-score	support
0	0.91	0.90	0.91	314
1	0.74	0.76	0.75	114
accuracy			0.87	428
macro avg	0.83	0.83	0.83	428
weighted avg	0.87	0.87	0.87	428

Tối ưu hóa siêu tham số

Tạo một scorer bằng cách sử dụng hàm `make_scorer` từ thư viện `sklearn`. Ở đây, scorer được tạo để tính toán `recall_score` với nhãn positive (`pos_label=1`). Recall là tỷ lệ các trường hợp positive được dự đoán chính xác trên tổng số các trường hợp positive thực tế.

Định nghĩa một từ điển `param_grid` chứa các tham số và giá trị mà `GridSearchCV` sẽ duyệt qua để tìm kiếm các siêu tham số tốt nhất cho mô hình `RandomForest`. Các tham số bao gồm '`n_estimators`' (số lượng cây trong mô hình), '`max_depth`' (độ sâu tối đa của các cây), '`min_samples_split`' (số lượng mẫu tối thiểu để chia một nút trong cây), '`min_samples_leaf`' (số lượng mẫu tối thiểu để tạo một lá trong cây), và '`bootstrap`' (có sử dụng bootstrap sampling hay không).

Tạo một đối tượng `GridSearchCV` từ thư viện `sklearn`. Đối số estimator là `rf_model` (mô hình `RandomForest` đã được khởi tạo trước đó), `param_grid` là từ điển các tham số và giá trị, `cv=5` chỉ định số lượng fold trong cross-validation (ở đây là 5), `verbose=2` để hiển thị thông tin chi tiết trong quá trình tìm kiếm các tham số và `scoring=scorer` để sử dụng scorer đã được định nghĩa trước đó để đánh giá mô hình.

Sử dụng phương thức `fit` của `GridSearchCV` để thực hiện tìm kiếm lưới (grid search) trên các tham số và giá trị trong `param_grid`. `X_train` là ma trận dữ liệu đặc trưng huấn luyện và `y_train` là vector nhãn tương ứng. Quá trình này sẽ thực hiện cross-validation với số lượng fold đã được chỉ định và tìm ra bộ tham số tốt nhất cho mô hình `RandomForest` dựa trên scorer đã được định nghĩa trước đó.

```
scorer = make_scorer(recall_score, pos_label=1)
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
```

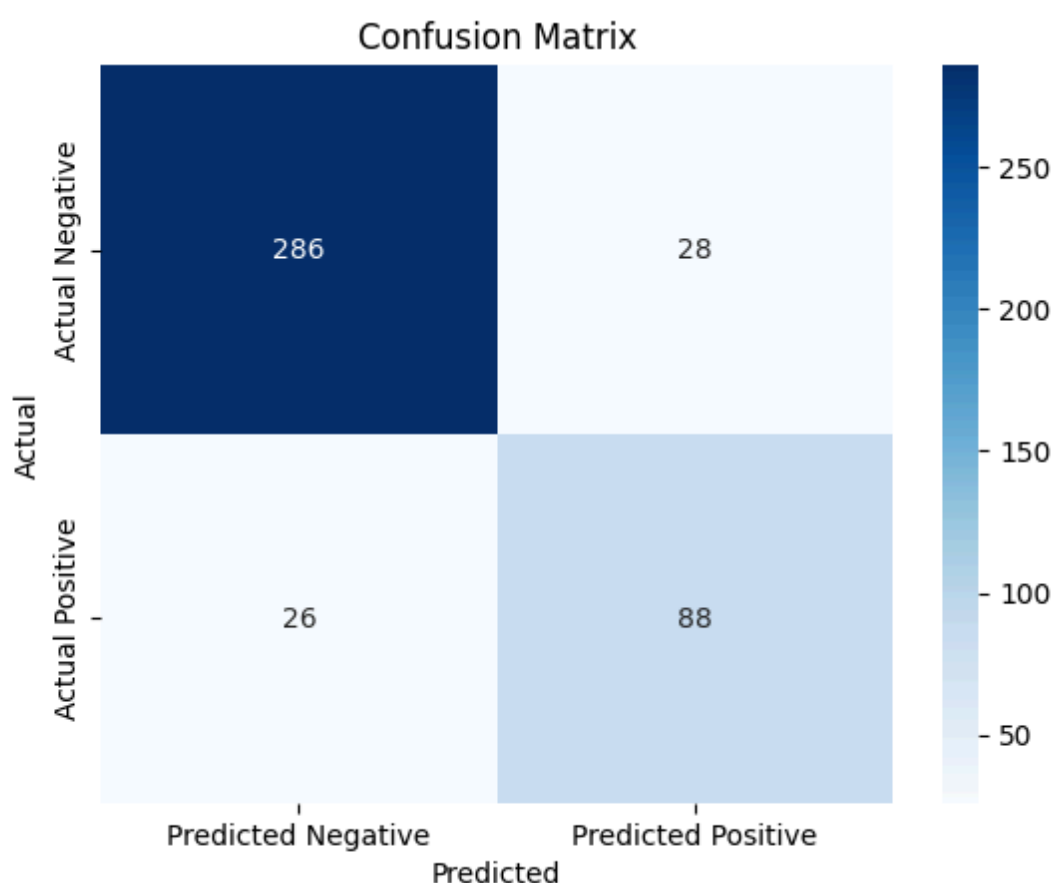
```
}  
grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid,  
                           cv=5, verbose=2, scoring=scorer)  
grid_search.fit(X_train, y_train)
```

Dòng code `y_pred = grid_search.best_estimator_.predict(X_test)` sử dụng mô hình tốt nhất tìm được từ quá trình grid search (`grid_search.best_estimator_`) để dự đoán nhãn cho dữ liệu `X_test`. Kết quả dự đoán được lưu vào biến `y_pred`.

```
y_pred = grid_search.best_estimator_.predict(X_test)  
đánh_gia(y_test, y_pred)
```

Độ chính xác: 0.8738317757009346
 Báo cáo phân loại:

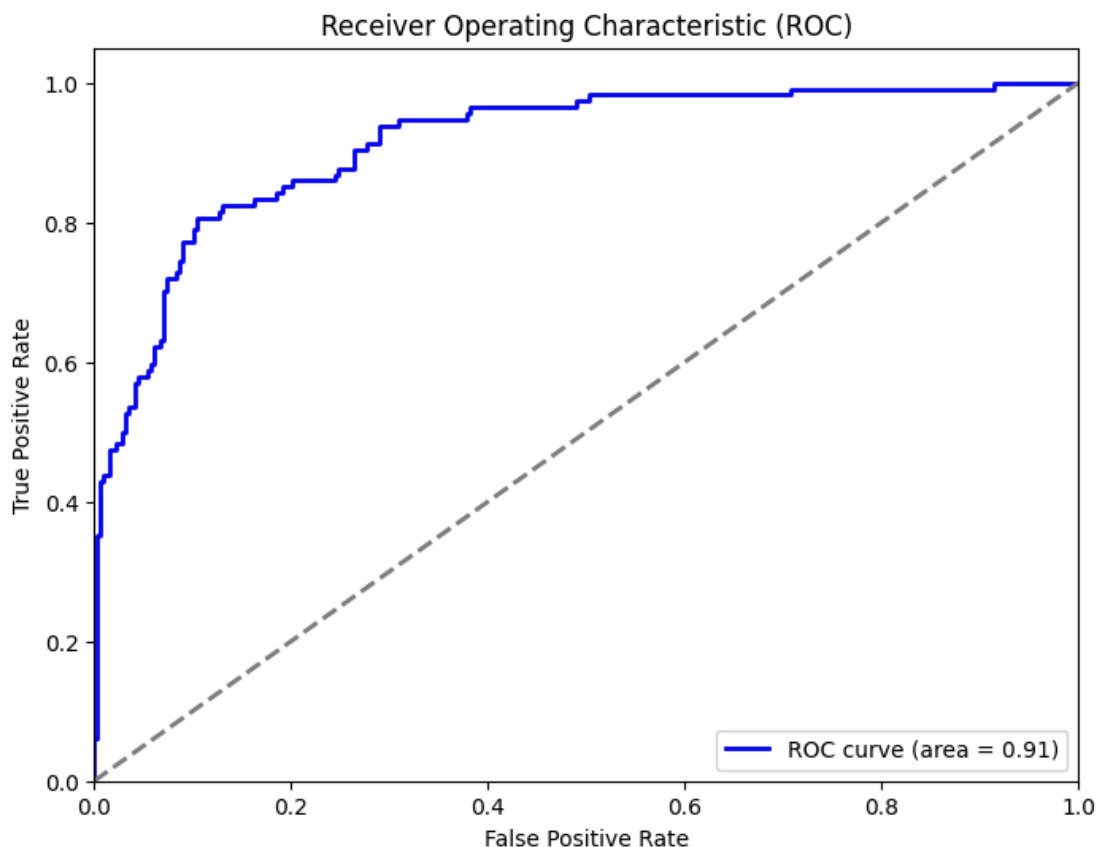
	precision	recall	f1-score	support
0	0.92	0.91	0.91	314
1	0.76	0.77	0.77	114
accuracy			0.87	428
macro avg	0.84	0.84	0.84	428
weighted avg	0.87	0.87	0.87	428



Nhận xét: Ta thấy, phương pháp tìm siêu tham số đã cải thiện được độ chính xác, cũng như các đánh giá về precision, recall và f1-score cũng được cải thiện trên nhãn dương tính (nhãn 1).

Thực hiện việc vẽ đường cong ROC (Receiver Operating Characteristic) để đánh giá hiệu suất của mô hình trong phân loại nhị phân.

```
plot_roc(grid_search.best_estimator_, X_test, y_test)
```



Chọn lọc đặc trưng

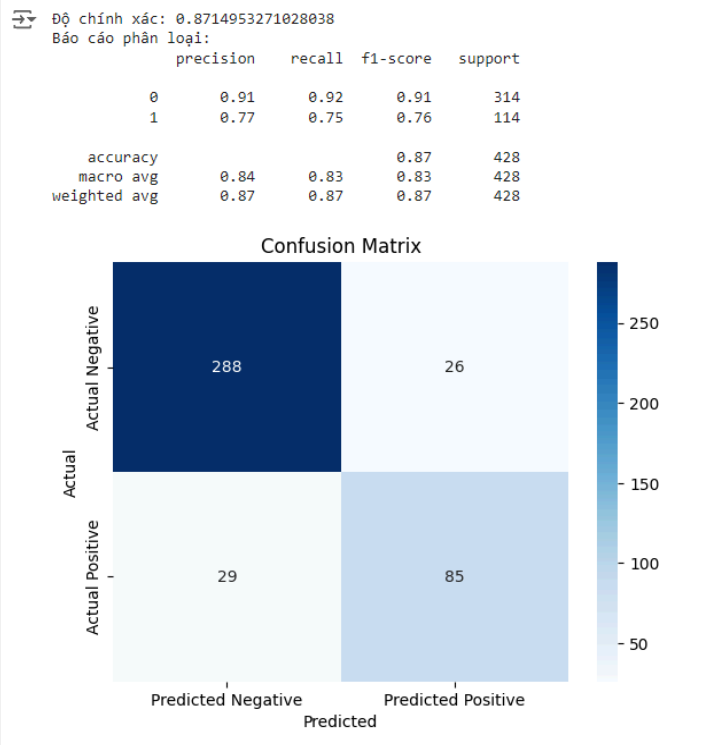
- Tạo một đối tượng RFE (Recursive Feature Elimination) từ thư viện sklearn. Đối số `rf_model` là mô hình RandomForest đã khởi tạo trước đó, và `n_features_to_select=0.90` chỉ định số lượng đặc trưng (features) sẽ được chọn sau quá trình loại bỏ đặc trưng lặp lại.
- Sử dụng phương thức `fit` của RFE để thực hiện quá trình loại bỏ đặc trưng lặp lại trên dữ liệu huấn luyện. `X_train` là ma trận dữ liệu đặc trưng huấn luyện và `y_train` là vector nhãn tương ứng.
- Sử dụng phương thức `transform` của RFE để giữ lại chỉ các đặc trưng được chọn sau quá trình loại bỏ. Kết quả được lưu vào biến `X_train_rfe`.
- Sử dụng phương thức `transform` của RFE để giữ lại chỉ các đặc trưng được chọn sau quá trình loại bỏ trên dữ liệu kiểm tra. Kết quả được lưu vào biến `X_test_rfe`.
- Tạo một đối tượng RandomForestClassifier mới từ thư viện sklearn. Đối số `random_state=42` đặt seed cho việc tạo ngẫu nhiên cây trong mô hình để đảm bảo kết quả tái lập được.
- Sử dụng phương thức `fit` của model để huấn luyện mô hình RandomForest trên dữ liệu huấn luyện sau khi đã loại bỏ đặc trưng lặp lại. `X_train_rfe` là ma trận dữ liệu đặc trưng huấn luyện đã được chọn sau quá trình loại bỏ đặc trưng lặp lại và `y_train` là vector nhãn tương ứng.

- Sử dụng phương thức predict của model để dự đoán nhãn cho dữ liệu X_test_rfe (dữ liệu kiểm tra sau khi đã loại bỏ đặc trưng lặp lại). Kết quả dự đoán được lưu vào biến y_pred.

```
rfe = RFE(rf_model, n_features_to_select=0.90)
rfe.fit(X_train, y_train)

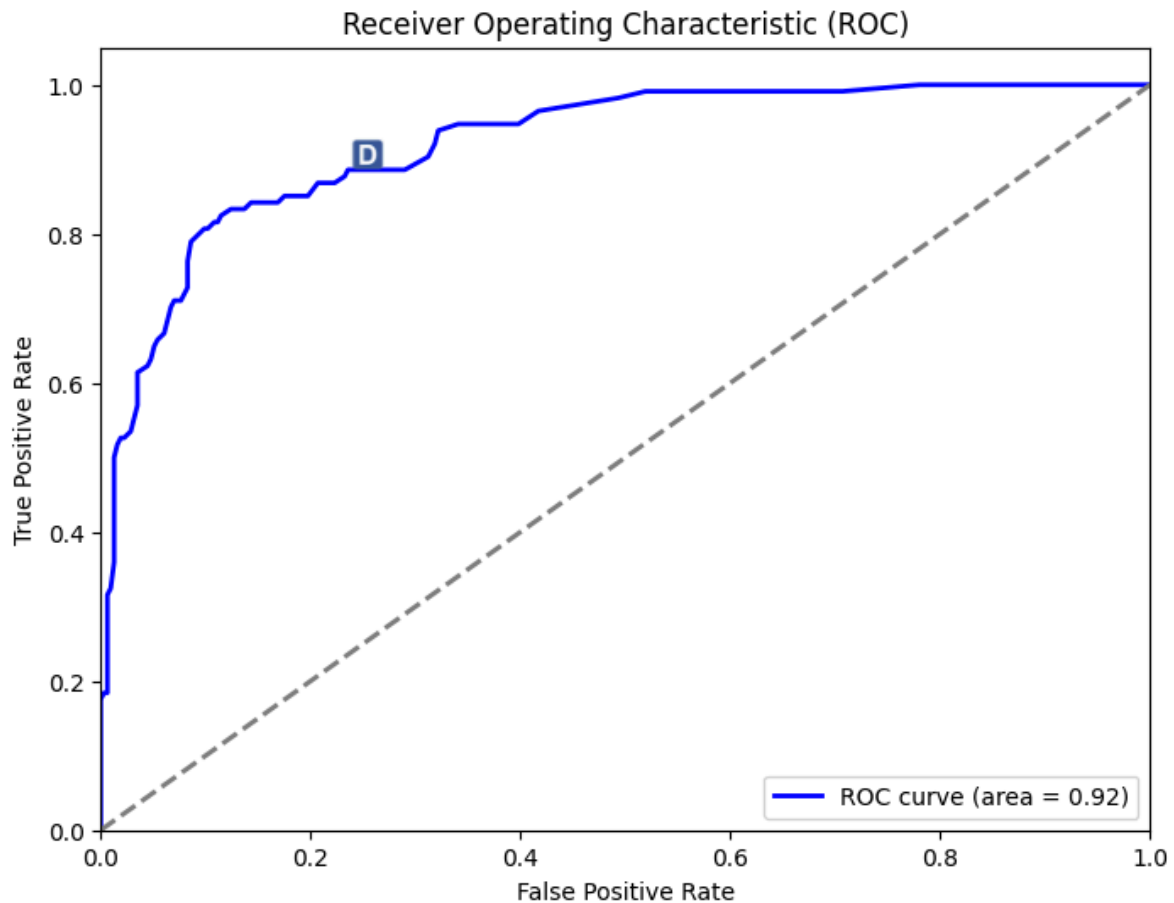
X_train_rfe = rfe.transform(X_train)
X_test_rfe = rfe.transform(X_test)

rf_rfe = RandomForestClassifier(random_state=42)
rf_rfe.fit(X_train_rfe, y_train)
y_pred = rf_rfe.predict(X_test_rfe)
danh_gia(y_test, y_pred)
```



- **Nhận xét:**
 - Độ chính xác tăng nhẹ.
 - Độ nhạy recall ở nhãn âm tính tăng nhẹ

- Precision ở nhãn dương tính tăng nhẹ, nhưng giá trị recall ở nhãn dương tính giảm nhẹ.
- Thực hiện việc vẽ đường cong ROC (Receiver Operating Characteristic) để đánh giá hiệu suất của mô hình trong phân loại nhị phân.



4.4.2. Gradient Boost Classifier

- Tạo một đối tượng GradientBoostingClassifier từ thư viện sklearn và gán cho biến gbc_model.
- Phương thức fit được gọi để huấn luyện mô hình Gradient Boosting trên dữ liệu huấn luyện X_train và y_train.

```
gbc_model = GradientBoostingClassifier()
gbc_model.fit(X_train, y_train)
```

- Sử dụng mô hình Gradient Boosting đã huấn luyện (gbc_model) để dự đoán nhãn cho dữ liệu X_test. Kết quả dự đoán được lưu vào biến y_pred.

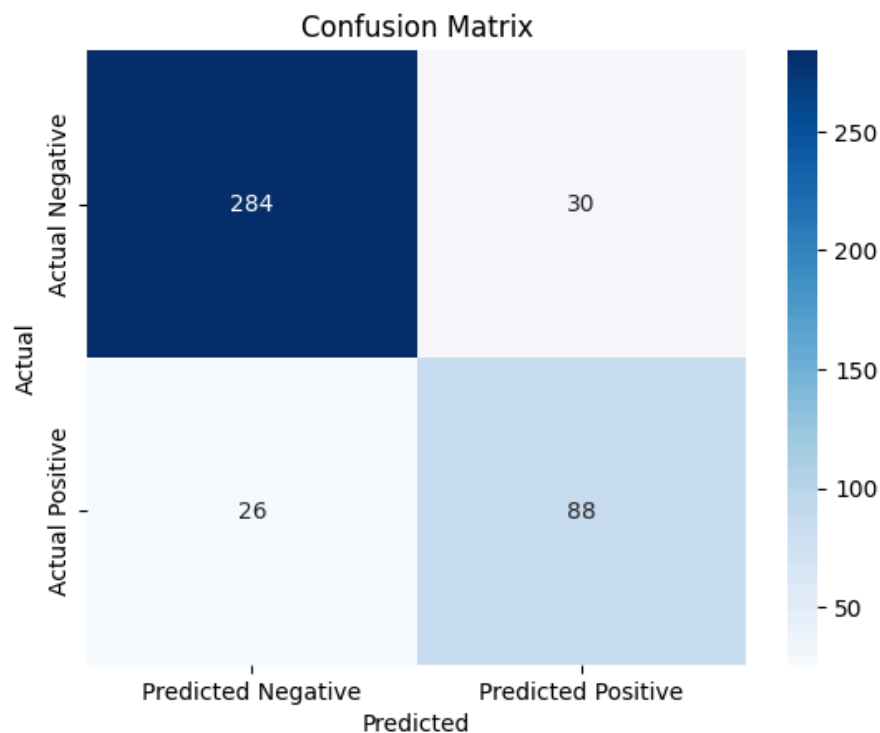
```
y_pred = gbc_model.predict(X_test)
```

```
danh_gia(y_test, y_pred)
```

Độ chính xác: 0.8691588785046729

Báo cáo phân loại:

	precision	recall	f1-score	support
0	0.92	0.90	0.91	314
1	0.75	0.77	0.76	114
accuracy			0.87	428
macro avg	0.83	0.84	0.83	428
weighted avg	0.87	0.87	0.87	428



```
df.head()
```

	time	trt	age	wtkg	hemo	homo	drugs	karnof	oprior	z30	...	str2	strat	symptom	treat	offtrt	cd40	cd420	cd80	cd820	infected
0	948	2	48	89.8128	0	0	0	100	0	0	...	0	1	0	1	0	422	477	566	324	0
1	1002	3	61	49.4424	0	0	0	90	0	1	...	1	3	0	1	0	162	218	392	564	1
2	961	3	45	88.4520	0	1	1	90	0	1	...	1	3	0	1	1	326	274	2063	1893	0
3	1166	3	47	85.2768	0	1	0	100	0	1	...	1	3	0	1	0	287	394	1590	966	0
4	1090	0	43	66.6792	0	1	0	100	0	1	...	1	3	0	0	0	504	353	870	782	0

5 rows x 23 columns

Tối ưu hóa siêu tham số

Các tham số trong `gbc_grid` được sử dụng để tìm kiếm siêu tham số (hyperparameters) tốt nhất cho mô hình Gradient Boosting. Dưới đây là giải thích cho các tham số được sử dụng trong `gbc_grid`:

- `n_estimators`: Số lượng cây quyết định trong ensemble. Đây là số lượng cây được huấn luyện trong quá trình Gradient Boosting.
- `learning_rate`: Tỷ lệ học (learning rate) điều chỉnh đóng góp của mỗi cây trong ensemble. Giá trị nhỏ hơn sẽ tạo ra một tốc độ học chậm và giúp mô hình hội tụ tốt hơn, nhưng cũng có thể yêu cầu số lượng cây lớn hơn để đạt được hiệu suất tối ưu.
- `max_depth`: Số lượng tối đa các nút trên một cây quyết định. Giới hạn độ sâu của các cây trong ensemble. Điều này giúp kiểm soát độ phức tạp của mô hình và tránh việc quá khớp (overfitting).
- `min_samples_split`: Số lượng mẫu tối thiểu yêu cầu để một nút có thể được chia thành hai nút con trong quá trình xây dựng cây. Giới hạn kích thước tối thiểu của các nút để tránh việc chia nhỏ quá.
- `min_samples_leaf`: Số lượng mẫu tối thiểu yêu cầu để một nút được coi là lá (leaf node). Giới hạn kích thước tối thiểu của các lá để tránh việc tạo ra các lá với số lượng mẫu quá nhỏ.
- `subsample`: Tỷ lệ mẫu con (subsample) được sử dụng để huấn luyện từng cây. Giá trị nhỏ hơn 1.0 cho phép sử dụng một phần của dữ liệu huấn luyện ngẫu nhiên cho mỗi cây, giúp giảm sự phụ thuộc giữa các cây.
- `max_features`: Số lượng đặc trưng tối đa được xem xét khi tìm kiếm phân chia tốt nhất cho mỗi nút. Có thể là "sqrt" (căn bậc hai của số lượng đặc trưng), "log2" (logarit cơ số 2 của số lượng đặc trưng) hoặc một số nguyên nhỏ.

Tạo một đối tượng `GridSearchCV` từ thư viện `sklearn` và gán cho biến `grid_search`. Đối số estimator được đặt là `gbc_model`, tức là mô hình Gradient Boosting đã được khởi tạo trước đó. `param_grid` là một từ điển (dictionary) chứa các tham số và giá trị để tìm kiếm trong quá trình grid search.

```
gbc_grid = {  
    'n_estimators': [100, 200, 300],  
    'learning_rate': [0.01, 0.05, 0.1],  
    'max_depth': [3, 4, 5],  
    'min_samples_split': [2, 5, 10],  
    'min_samples_leaf': [1, 2, 4],  
}
```



```
'subsample': [0.8, 0.9, 1.0],
'max_features': ['sqrt', 'log2']
}

grid_search = GridSearchCV(estimator=gbc_model, param_grid=gbc_grid, cv=3,
verbose=2)

grid_search.fit(X_train, y_train)
```

- `grid_search.best_params_` sẽ trả về một từ điển (dictionary) chứa các tham số tốt nhất tìm thấy trong quá trình tìm kiếm lưới. Mỗi khóa trong từ điển tương ứng với tên của một tham số và giá trị tương ứng là giá trị tốt nhất tìm thấy cho tham số đó.

```
grid_search.best_params_
```

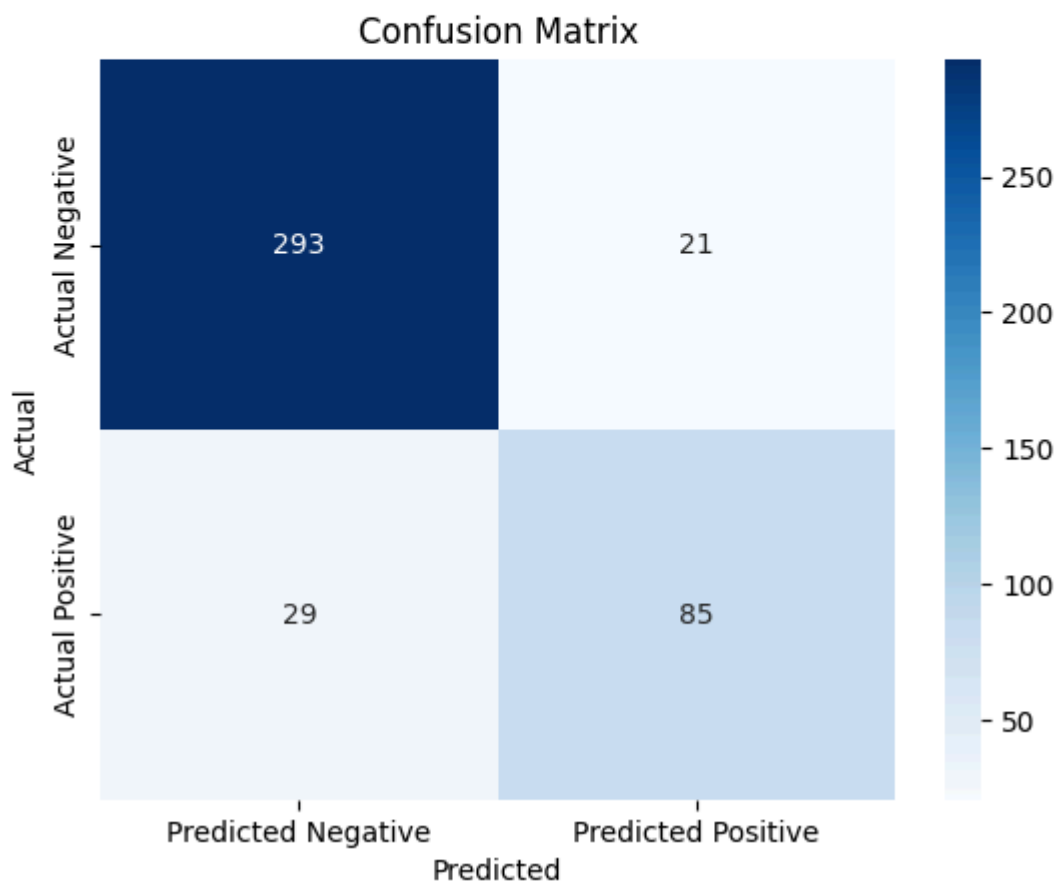
```
{'learning_rate': 0.1,
 'max_depth': 5,
 'max_features': 'log2',
 'min_samples_leaf': 4,
 'min_samples_split': 10,
 'n_estimators': 300,
 'subsample': 1.0}
```

- Sử dụng mô hình tốt nhất đã tìm thấy từ quá trình tìm kiếm lưới (`grid_search`) để dự đoán nhãn cho dữ liệu `X_test`. Kết quả dự đoán được lưu vào biến `y_pred`.

Độ chính xác: 0.883177570093458

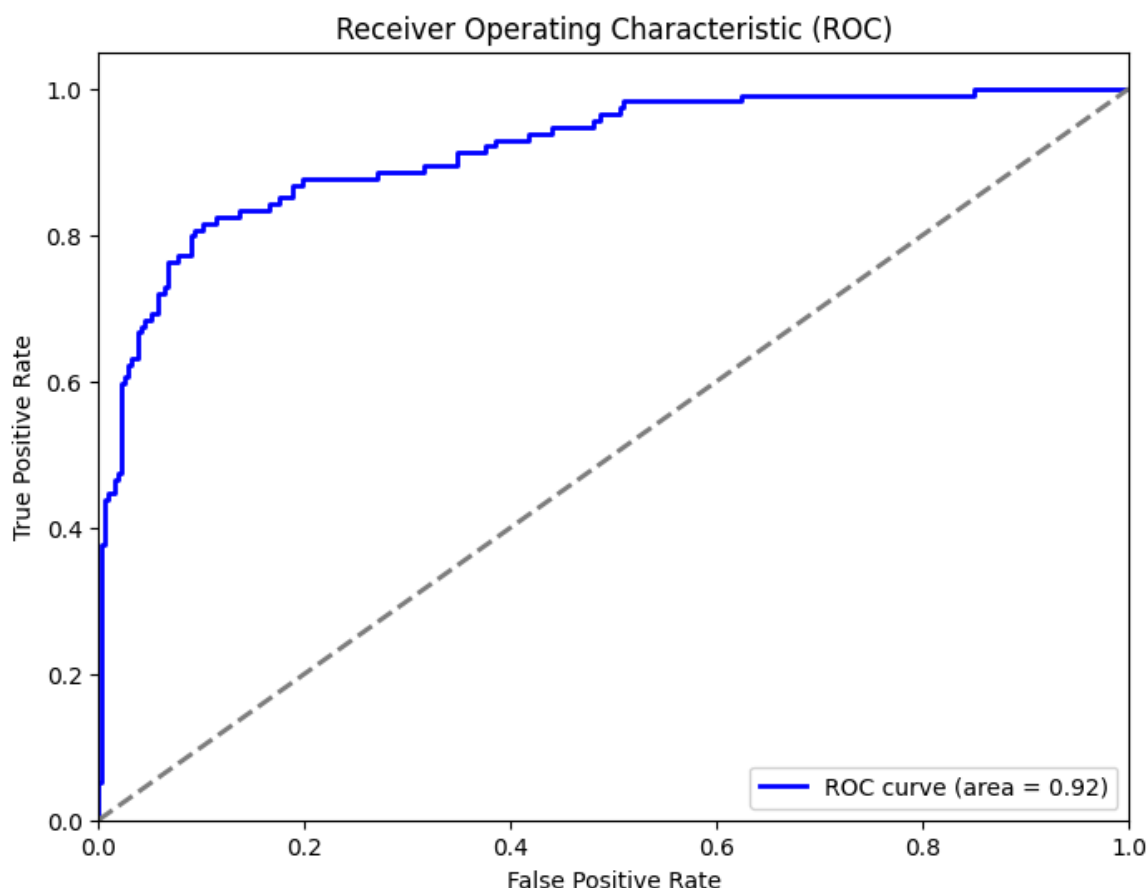
Báo cáo phân loại:

	precision	recall	f1-score	support
0	0.91	0.93	0.92	314
1	0.80	0.75	0.77	114
accuracy			0.88	428
macro avg	0.86	0.84	0.85	428
weighted avg	0.88	0.88	0.88	428



- Vẽ đường cong ROC cho mô hình tốt nhất đã tìm thấy từ quá trình tìm kiếm lưới (grid_search)

```
plot_roc(grid_search.best_estimator_, X_test, y_test)
```



4.4.3. XG Boost Classifier

Tạo mô hình XGBoost và huấn luyện trên dữ liệu huấn luyện X_{train} và y_{train} . Dưới đây là giải thích cho các tham số và phương thức được sử dụng:

- **XGBClassifier**: Đây là lớp mô hình XGBoost Classifier trong thư viện XGBoost. Nó được sử dụng để tạo mô hình XGBoost cho bài toán phân loại.
- **use_label_encoder=False**: Tham số này chỉ định rằng không sử dụng trình mã hóa nhãn mặc định của XGBoost. Nếu nhãn của bạn đã được mã hóa trước đó (ví dụ: thành số nguyên), bạn có thể đặt tham số này thành False.
- **eval_metric='logloss'**: Tham số này xác định phương pháp đánh giá hiệu suất của mô hình trong quá trình huấn luyện. Trong trường hợp này, đánh giá được dùng là logloss, tức là log-loss function. Log-loss là một phương pháp đo lường độ chính xác của mô hình dự đoán xác suất trong bài toán phân loại.

Phương thức `fit(X_{train} , y_{train})` được gọi để huấn luyện mô hình XGBoost trên dữ liệu (X_{train} , y_{train}).

```
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss')
xgb_model.fit(X_train, y_train)
```

Sử dụng mô hình XGBoost đã được huấn luyện (xgb_model) để dự đoán nhãn cho dữ liệu X_test. Kết quả dự đoán được lưu vào biến y_pred.

```
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss')
xgb_model.fit(X_train, y_train)
```

Tối ưu hóa siêu tham số

Tạo một từ điển (dictionary) chứa các tham số và giá trị tương ứng mà chúng ta muốn thử trong quá trình tìm kiếm lưới. Các tham số bao gồm:

- **n_estimators**: Đây là số lượng cây quyết định (decision trees) trong mô hình XGBoost. Nó xác định số lượng vòng lặp (iterations) mà thuật toán XGBoost sẽ thực hiện để huấn luyện mô hình. Giá trị này cần được điều chỉnh để đảm bảo mô hình đạt hiệu suất tốt nhưng không gây overfitting.
- **learning_rate**: Tham số này xác định tỷ lệ học (learning rate) trong thuật toán XGBoost. Nó quyết định mức độ tác động của mỗi cây quyết định đối với mô hình cuối cùng. Giá trị của learning rate cần được điều chỉnh cẩn thận, vì một learning rate quá cao có thể dẫn đến việc mô hình không hội tụ hoặc overfitting.
- **max_depth**: Đây là độ sâu tối đa của các cây quyết định trong mô hình XGBoost. Nó xác định số lượng lớp quyết định tối đa mà cây có thể có. Điều chỉnh max_depth có thể ảnh hưởng đến tỷ lệ overfitting và underfitting của mô hình.
- **subsample**: Tham số này xác định tỷ lệ mẫu (samples) được sử dụng để huấn luyện mỗi cây quyết định trong mô hình XGBoost. Giá trị nhỏ hơn 1.0 sẽ tạo ra một tập mẫu con ngẫu nhiên từ dữ liệu huấn luyện để giảm sự phụ thuộc giữa các cây quyết định và tránh overfitting.
- **colsample_bytree**: Đây là tỷ lệ số lượng đặc trưng (features) được sử dụng để huấn luyện mỗi cây quyết định trong mô hình XGBoost. Giá trị nhỏ hơn 1.0 sẽ tạo ra một tập các đặc trưng con ngẫu nhiên từ tất cả các đặc trưng để giảm sự phụ thuộc giữa các cây quyết định và tránh overfitting.
- **gamma**: Tham số này xác định mức cắt (threshold) để tạo ra sự chia nhánh (split) trong mô hình XGBoost. Nếu giá trị của một nút lá (leaf node) giảm tổn thất (loss) lớn hơn gamma, thì nút đó sẽ được chia nhánh. Gamma cung cấp một cách giới hạn để kiểm soát việc chia nhánh và giảm overfitting.

- `min_child_weight`: Đây là trọng số tối thiểu của một nút lá (leaf node) trong mô hình XGBoost. Nếu tổn thất (loss) tại một nút lá nhỏ hơn `min_child_weight`, thì quá trình chia nhánh sẽ dừng lại. Tham số này có thể được sử dụng để kiểm soát việc chia nhánh và giảm overfitting.

Tạo đối tượng `GridSearchCV`. Nó được sử dụng để thực hiện tìm kiếm lưới trên mô hình XGBoost. Các tham số chính của `GridSearchCV` được sử dụng là:

- `estimator`: Đối tượng mô hình XGBoost (`xgb_model`) mà chúng ta muốn tìm kiếm lưới trên.
- `param_grid`: Từ điển (dictionary) chứa các tham số và giá trị tương ứng mà chúng ta muốn thử.
- `scoring`: Phương thức đánh giá hiệu suất mô hình, ví dụ như accuracy, precision, recall, hoặc một metric tự định nghĩa.
- `verbose`: Cấp độ chi tiết của quá trình tìm kiếm lưới. Giá trị 3 được sử dụng ở đây để hiển thị thông tin chi tiết về quá trình tìm kiếm.
- `cv`: Số lượng fold trong cross-validation.

```
xgb_grid = {
    'n_estimators': [50, 100, 200, 300],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'gamma': [0, 0.1, 0.2],
    'min_child_weight': [1, 3, 5]
}
xgb_rand_search = GridSearchCV(estimator=xgb_model, param_grid=xgb_grid,
                               scoring=scorer, verbose=3, cv=5)
xgb_rand_search.fit(X_train, y_train)
```

Trả về một từ điển (dictionary) chứa các tham số tốt nhất tìm thấy trong quá trình tìm kiếm lưới

```
xgb_rand_search.best_params_
```

```
{'subsample': 1.0,  
  'n_estimators': 200,  
  'min_child_weight': 1,  
  'max_depth': 7,  
  'learning_rate': 0.01,  
  'gamma': 0,  
  'colsample_bytree': 0.6}
```

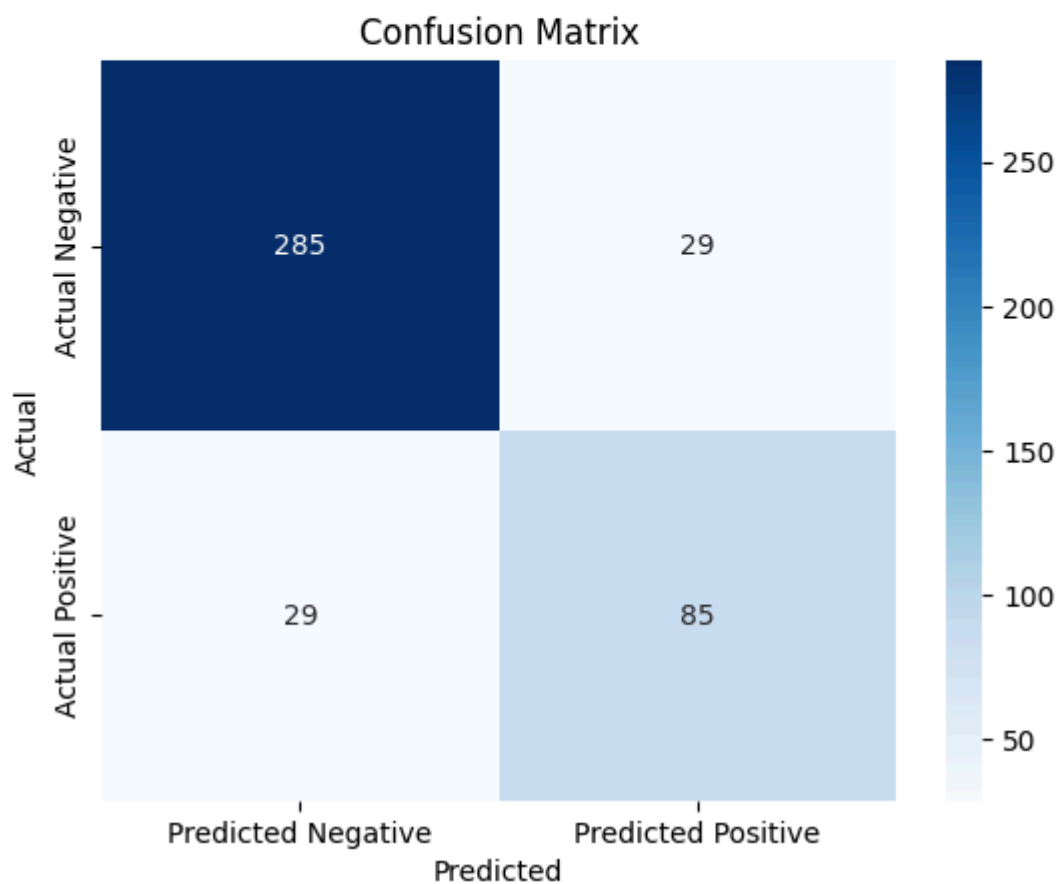
Sử dụng mô hình XGBoost đã được tìm kiếm ngẫu nhiên (xgb_rand_search) để dự đoán nhãn (y_pred) cho tập dữ liệu X_test

```
y_pred = xgb_rand_search.predict(X_test)  
đánh_gia(y_test, y_pred)
```

Độ chính xác: 0.8644859813084113

Báo cáo phân loại:




	precision	recall	f1-score	support
0	0.91	0.91	0.91	314
1	0.75	0.75	0.75	114
accuracy			0.86	428
macro avg	0.83	0.83	0.83	428
weighted avg	0.86	0.86	0.86	428



PHẦN KẾT LUẬN

- Trước khi cải tiến, mô hình cho kết quả tốt nhất là XG Boost Classifier.
- Sau cải tiến bằng cách tìm siêu tham số thông qua GridSearchCV, Gradient Boosting Classifier cho kết quả đánh giá nhỉnh hơn ở Accuracy, Precision, Random Forest Classifier cho kết quả Recall cao hơn.
- Mô hình Random Forest Classifier cho thấy sự cải thiện tốt hơn khi sử dụng GridSearchCV để tìm siêu tham số.
- Tuy nhiên, các sự cải thiện của hai mô sau không đạt được giá trị cao như XG Boost Classifier. Chính vì thế, mô hình tốt nhất trong 3 mô hình được xây dựng là XG Boost Classifier.

TÀI LIỆU THAM KHẢO

1. Oreilly, “Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow”.
2. Bui Tien Tung, “Gradient Boosting - Tất tần tât về thuật toán mạnh mẽ nhất trong Machine Learning”,
<https://viblo.asia/p/gradient-boosting-tat-tan-tat-ve-thuat-toan-manh-me-nhat-trong-machine-learning-YWOZrN7vZQ0>
3. AADARSH VELU, “ AIDS Classification EDA  + Acc 91% ”,
<https://www.kaggle.com/code/aadarshvelu/aids-classification-eda-acc-91>
4. MIFlow, “Automatic Logging with MLflow Tracking”,
<https://mlflow.org/docs/latest/tracking/autolog.html>

