

oooo



TRƯỜNG ĐẠI HỌC MỞ TP. HỒ CHÍ MINH
HO CHI MINH CITY OPEN UNIVERSITY

BÁO CÁO BÀI TẬP LỚN

MÔN HỌC: PHÂN TÍCH DỮ LIỆU

oooo

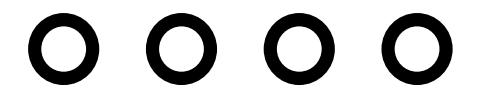
CHẨN ĐOÁN BỆNH TIỂU ĐƯỜNG DỰA VÀO BỘ
DỮ LIỆU KẾT QUẢ XÉT NGHIỆM MÁU

Giảng viên hướng dẫn: Hồ Hướng Thiên

Sinh viên thực hiện: Phạm Công Thuận - 2151013097

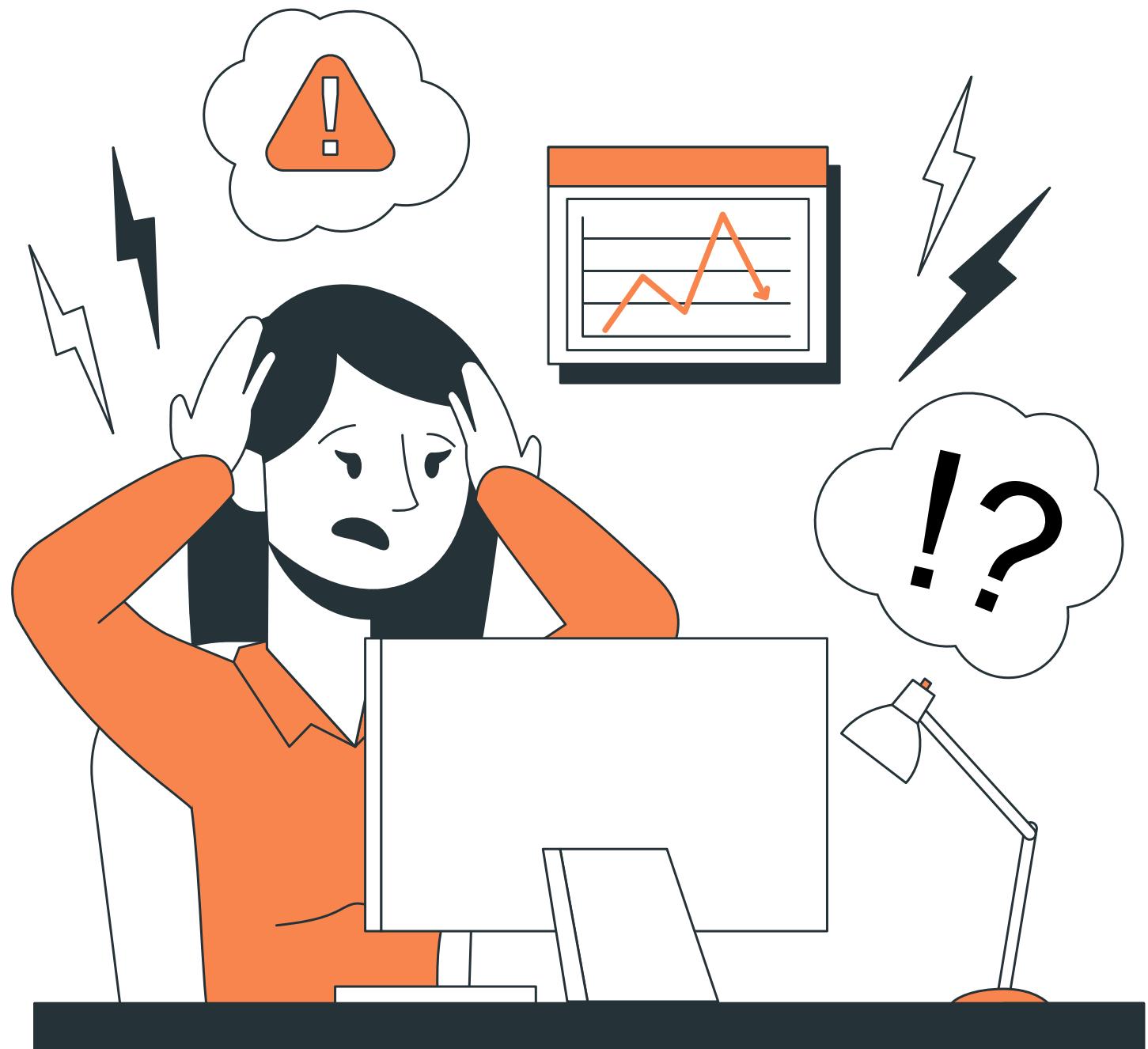
Tạ Thị Thiên Thanh - 2151013088





NỘI DUNG

1. Mô tả dữ liệu
2. Tiền xử lý dữ liệu
3. Gom cụm
4. Thuật toán phân lớp KNN
5. Thuật toán phân lớp Decision Tree
6. Kết luận



○ ○ ○ ○

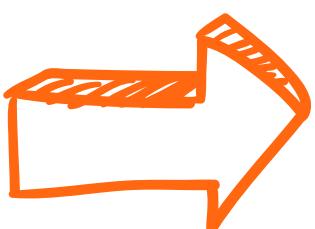
++ + + +

+ +
+ +
+ +
+ +
+ +

Mô tả dữ liệu

Bộ dữ liệu lâm sàng của hơn 5000 bệnh nhân

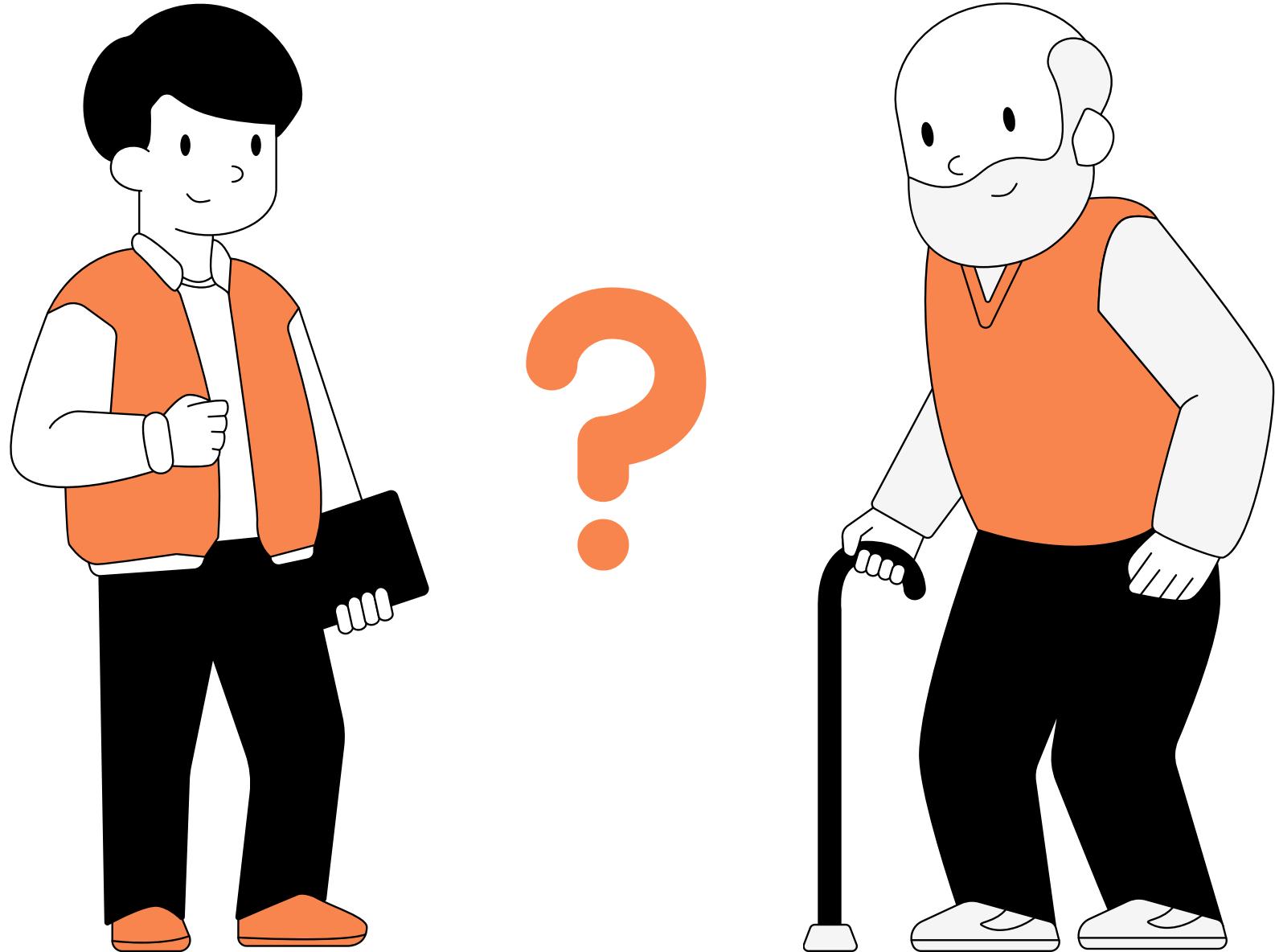
Phục vụ nghiên cứu chức năng thận, sức khỏe tim mạch, chẩn đoán tiểu đường



Đề tài tập trung nghiên cứu khả năng chẩn đoán bệnh tiểu đường

*Dữ liệu lâm sàng là dữ liệu thu được tại một cơ sở khám chữa bệnh chằng hạn như trạm y tế, phòng khám, bệnh viện

Thuộc tính ‘Age’



Số tuổi của bệnh nhân

Thuộc tính ‘Age’

mean	48.637853	Số tuổi trung bình
std	15.253447	Độ lệch chuẩn
min	-94.000000	Số tuổi nhỏ nhất
25%	36.000000	Tứ phân vị Q1
50%	49.000000	Giá trị trung vị (mean)
75%	59.000000	Tứ phân vị Q3
max	93.000000	Số tuổi cao nhất

Số lượng giá trị
không null



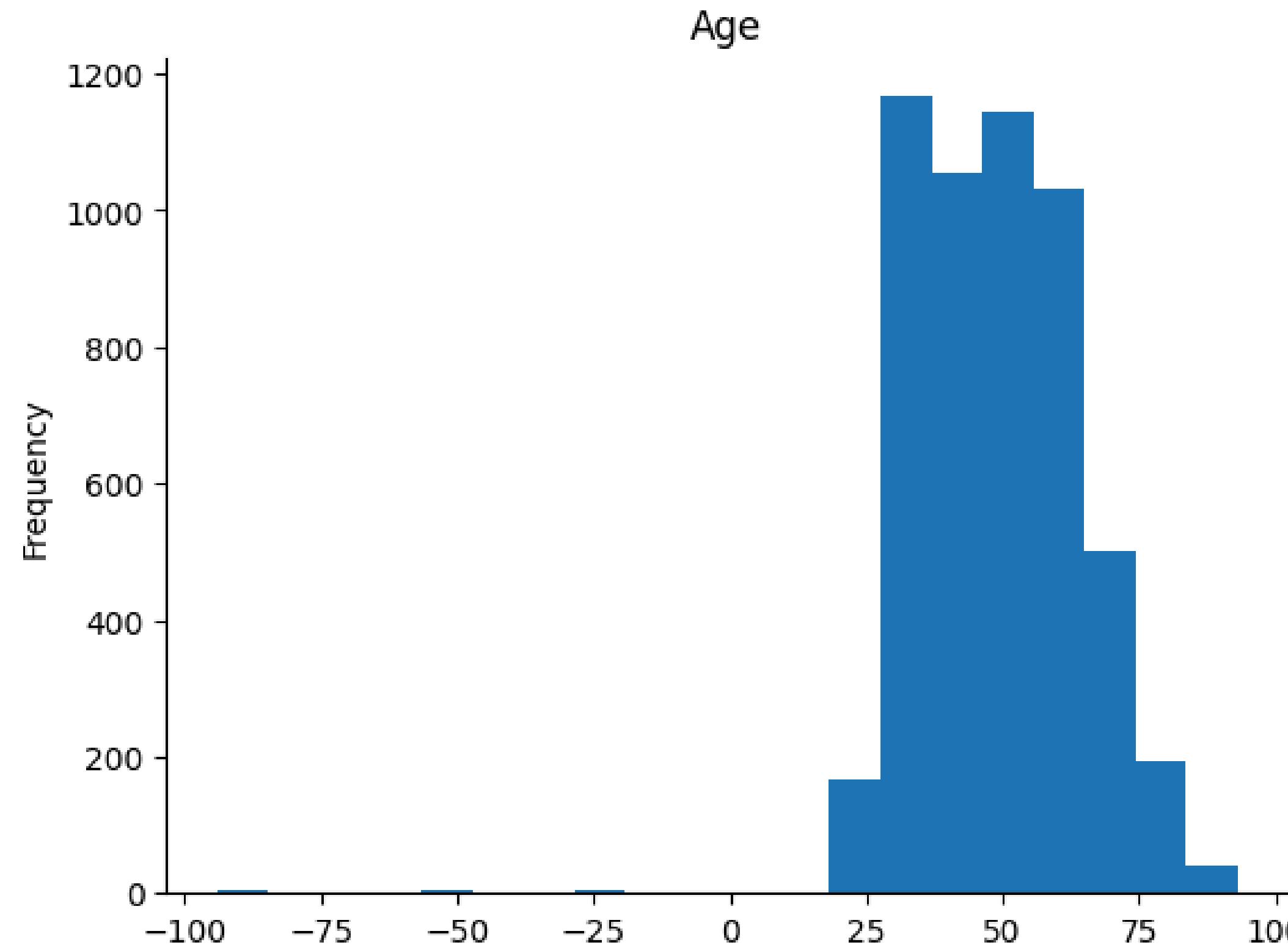
5310 non-null



float64

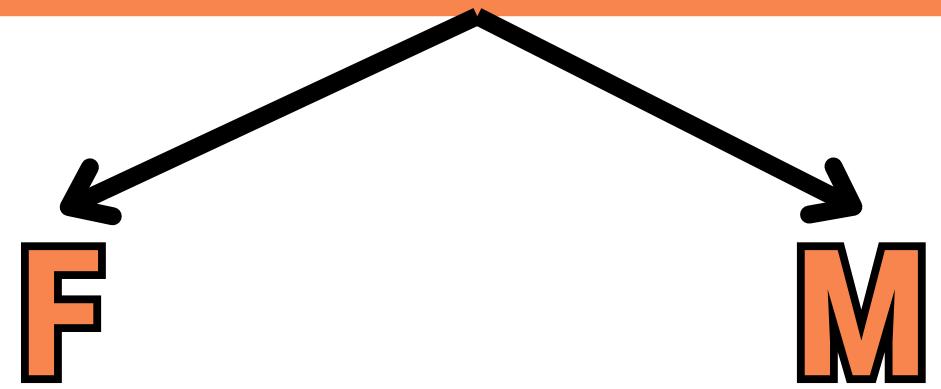
Kiểu dữ liệu
(datatype)

Thuộc tính ‘Age’

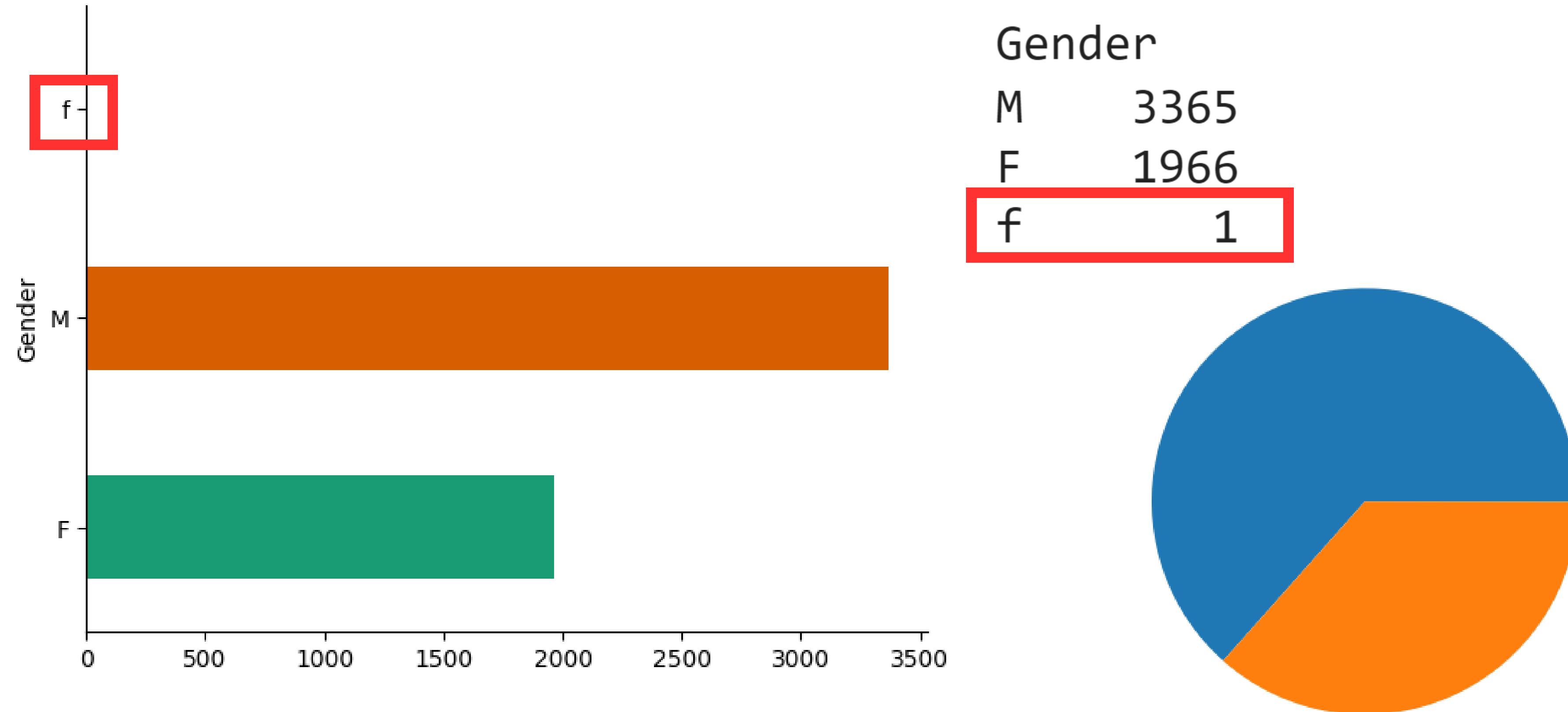


Thuộc tính ‘Gender’

Giới tính của bệnh nhân
bao gồm 2 giá trị

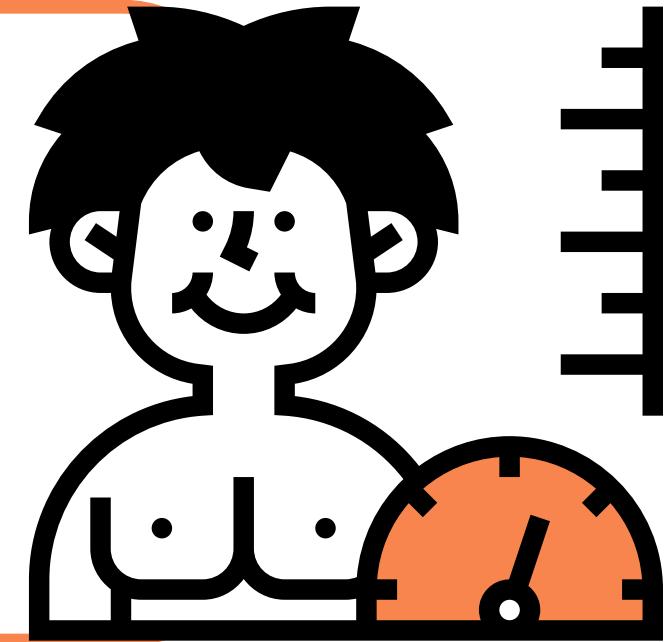


Thuộc tính ‘Gender’



Thuộc tính ‘BMI’

Body Mass Index (BMI): Chỉ số BMI của bệnh nhân.
BMI là chỉ số khối cơ thể dùng để xác định cân nặng
của một người đang thiếu cân, thừa cân hay cân đối



$$\frac{\text{weight}}{\text{height}^2}$$

Thuộc tính ‘BMI’

mean	24.130345	Giá trị BMI trung bình
std	7.718847	Độ lệch chuẩn
min	-99.000000	Giá trị BMI nhỏ nhất
25%	22.000000	Tứ phân vị Q1
50%	24.000000	Giá trị trung vị (mean)
75%	27.000000	Tứ phân vị Q3
max	47.000000	Giá trị BMI cao nhất

Số lượng giá trị
không null

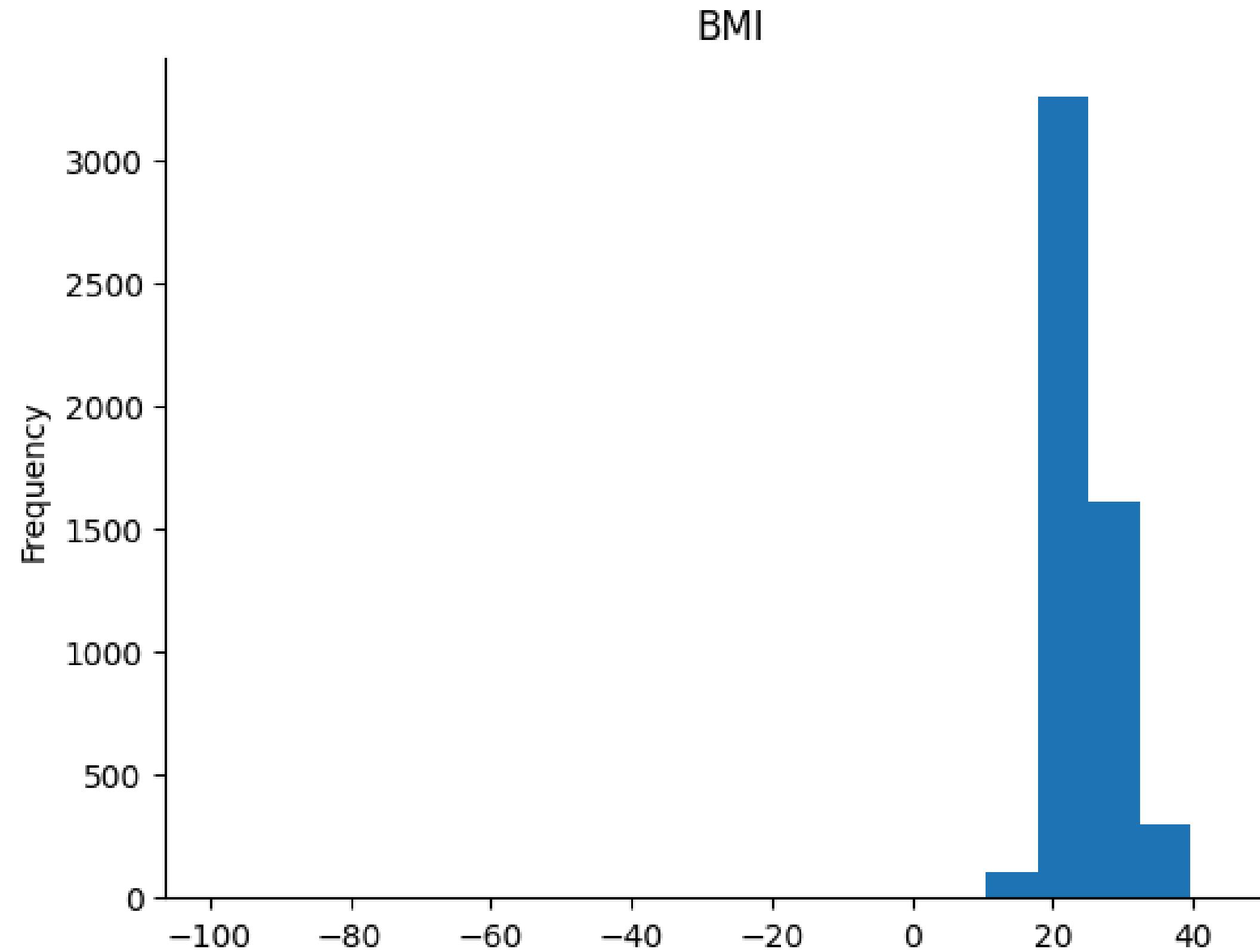


5309 non-null

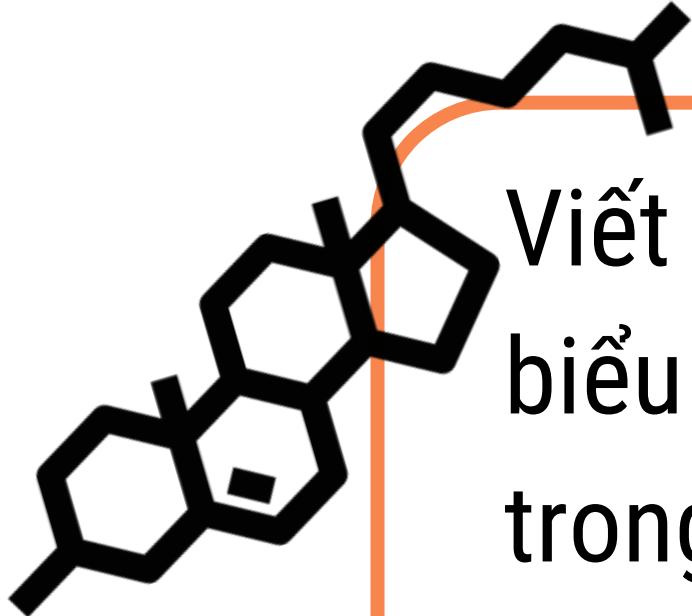
float64 →

Kiểu dữ liệu
(datatype)

Thuộc tính ‘BMI’



Thuộc tính ‘Chol’



Viết tắt của Cholesterol, biểu thị tỷ lệ cholesterol có trong máu.

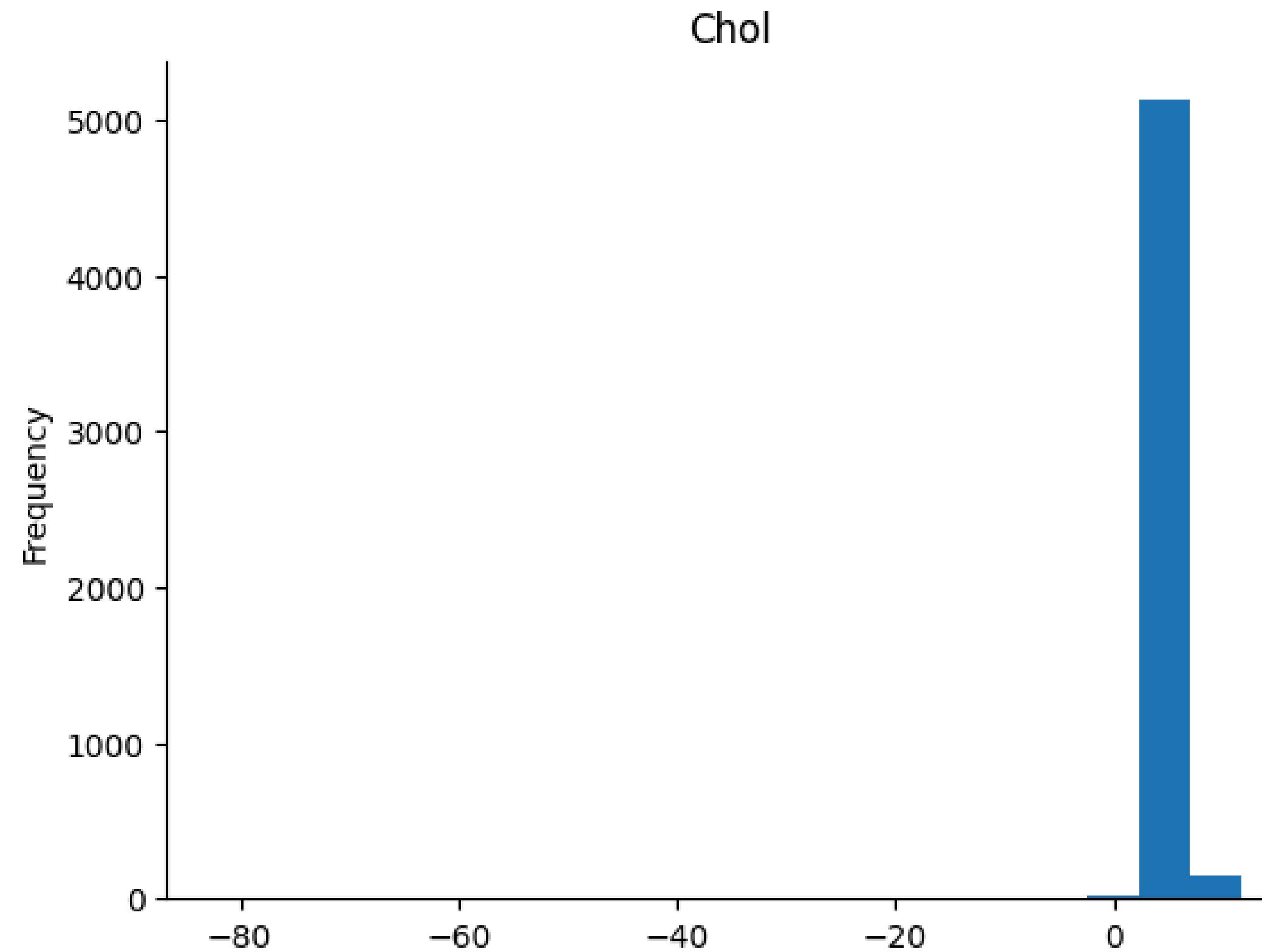
Cholesterol là một loại chất béo được sản sinh ra từ việc tiêu thụ thức ăn hoặc cơ thể tự sản xuất



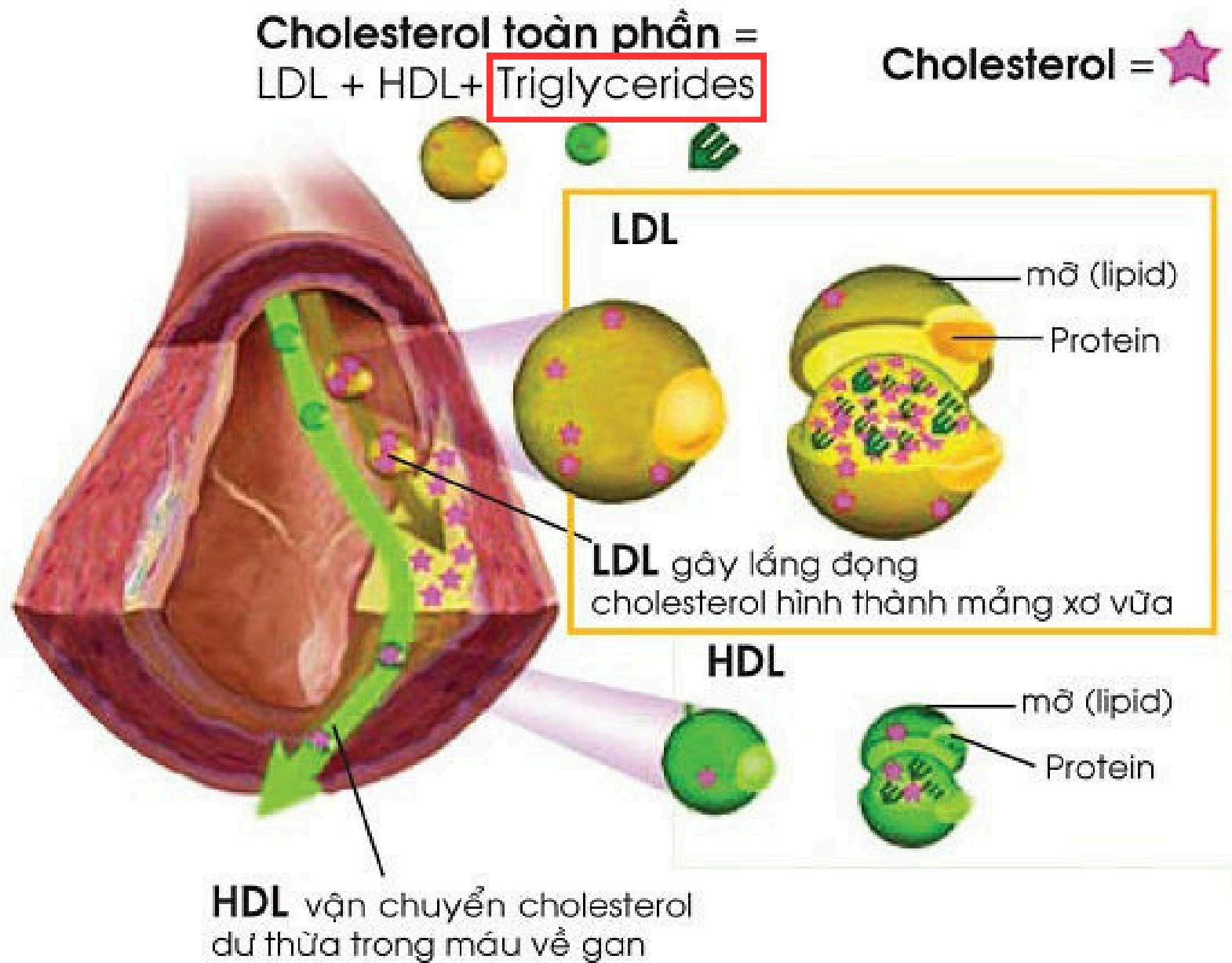
Thuộc tính ‘Chol’

mean	4.689629	Tỷ lệ Cholesterol trung bình
std	3.448396	Độ lệch chuẩn
min	-82.000000	Tỷ lệ Cholesterol thấp nhất
25%	4.180000	Tứ phân vị Q1
50%	4.800000	Giá trị trung vị (mean)
75%	5.460000	Tứ phân vị Q3
max	11.650000	Tỷ lệ Cholesterol cao nhất
Số lượng giá trị không null	← 5306 non-null →	Kiểu dữ liệu (datatype)

Thuộc tính ‘Chol’



Thuộc tính 'TG'



Là tỷ lệ triglycerides có trong máu.

Triglycerides là một dạng chất béo trung tính chứa 3 axit béo và có nguồn gốc từ mỡ động vật, thực vật mà bệnh nhân tiêu thụ.

CÁC THÀNH PHẦN MỠ MÁU

Thuộc tính ‘TG’

mean	1.503966	Tỷ lệ Triglycerides trung bình
std	3.983595	Độ lệch chuẩn
min	-94.000000	Tỷ lệ Triglycerides thấp nhất
25%	0.900000	Tứ phân vị Q1
50%	1.370000	Giá trị trung vị (mean)
75%	2.100000	Tứ phân vị Q3
max	32.640000	Tỷ lệ Triglycerides cao nhất

Số lượng giá trị
không null



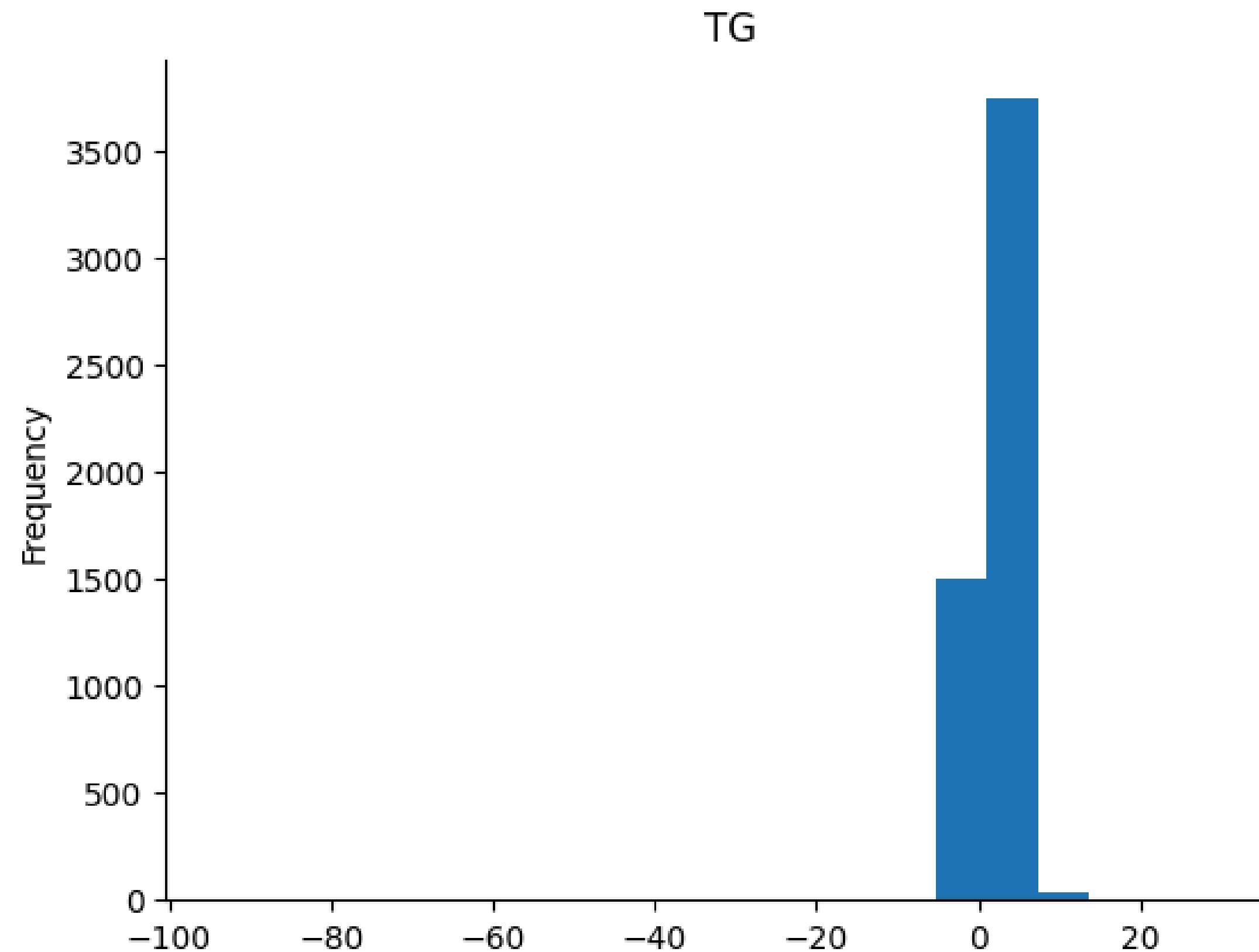
5300 non-null

float64

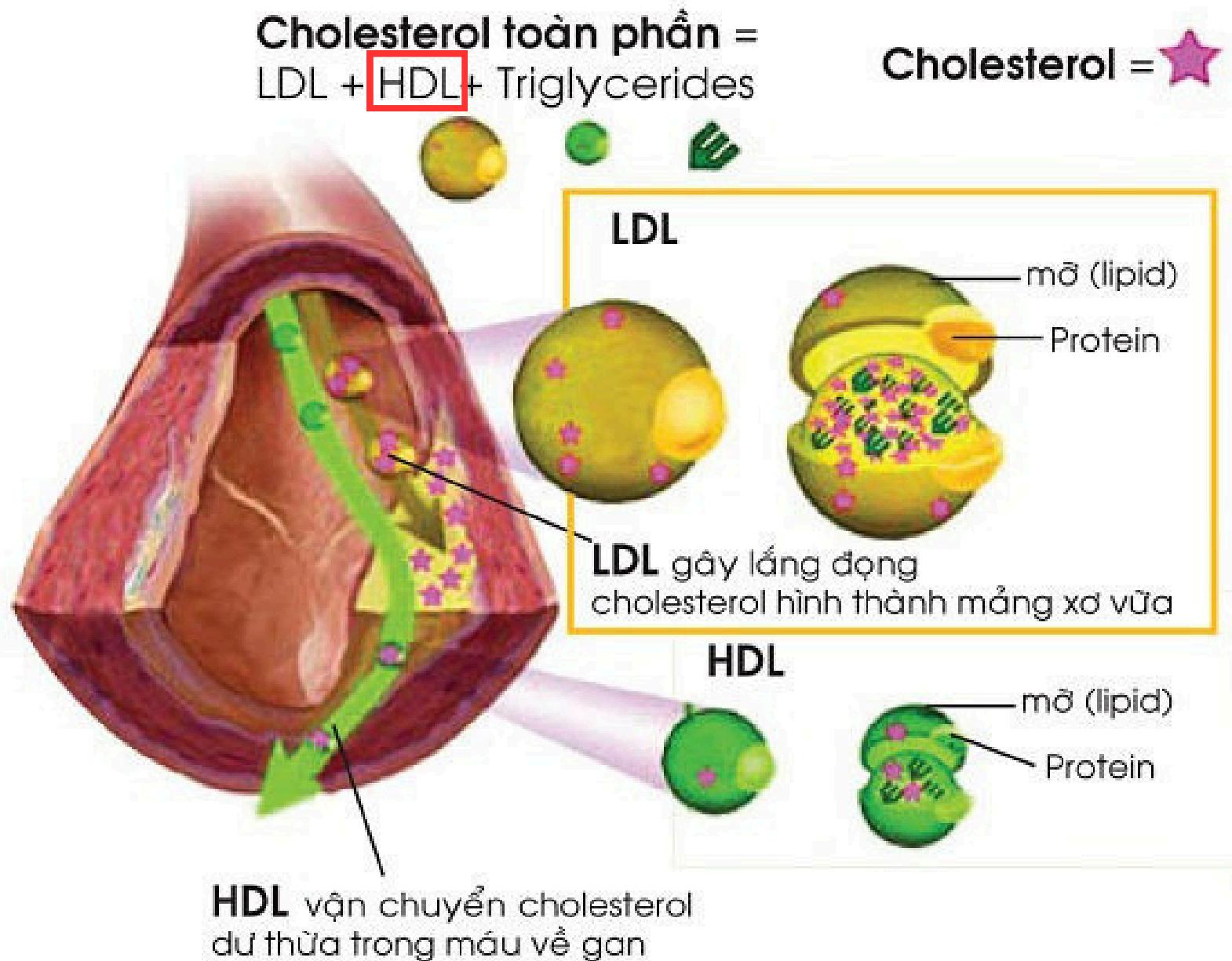


Kiểu dữ liệu
(datatype)

Thuộc tính 'TG'



Thuộc tính 'HDL'



CÁC THÀNH PHẦN MỠ MÁU

HDL (High-Density Lipoprotein) là chỉ số Lipoprotein tỷ trọng cao.

Lipoprotein tỷ trọng cao được xem như một loại cholesterol có lợi giúp vận chuyển cholesterol dư thừa tích trữ dưới mạch máu về gan để xử lý và đào thải ra ngoài.

Thuộc tính ‘HDL’

mean	1.346343	Chỉ số HDL trung bình
std	4.349107	Độ lệch chuẩn
min	-95.000000	Chỉ số HDL thấp nhất
25%	1.090000	Tứ phân vị Q1
50%	1.300000	Giá trị trung vị (mean)
75%	1.590000	Tứ phân vị Q3
max	9.900000	Chỉ số HDL cao nhất

Số lượng giá trị
không null

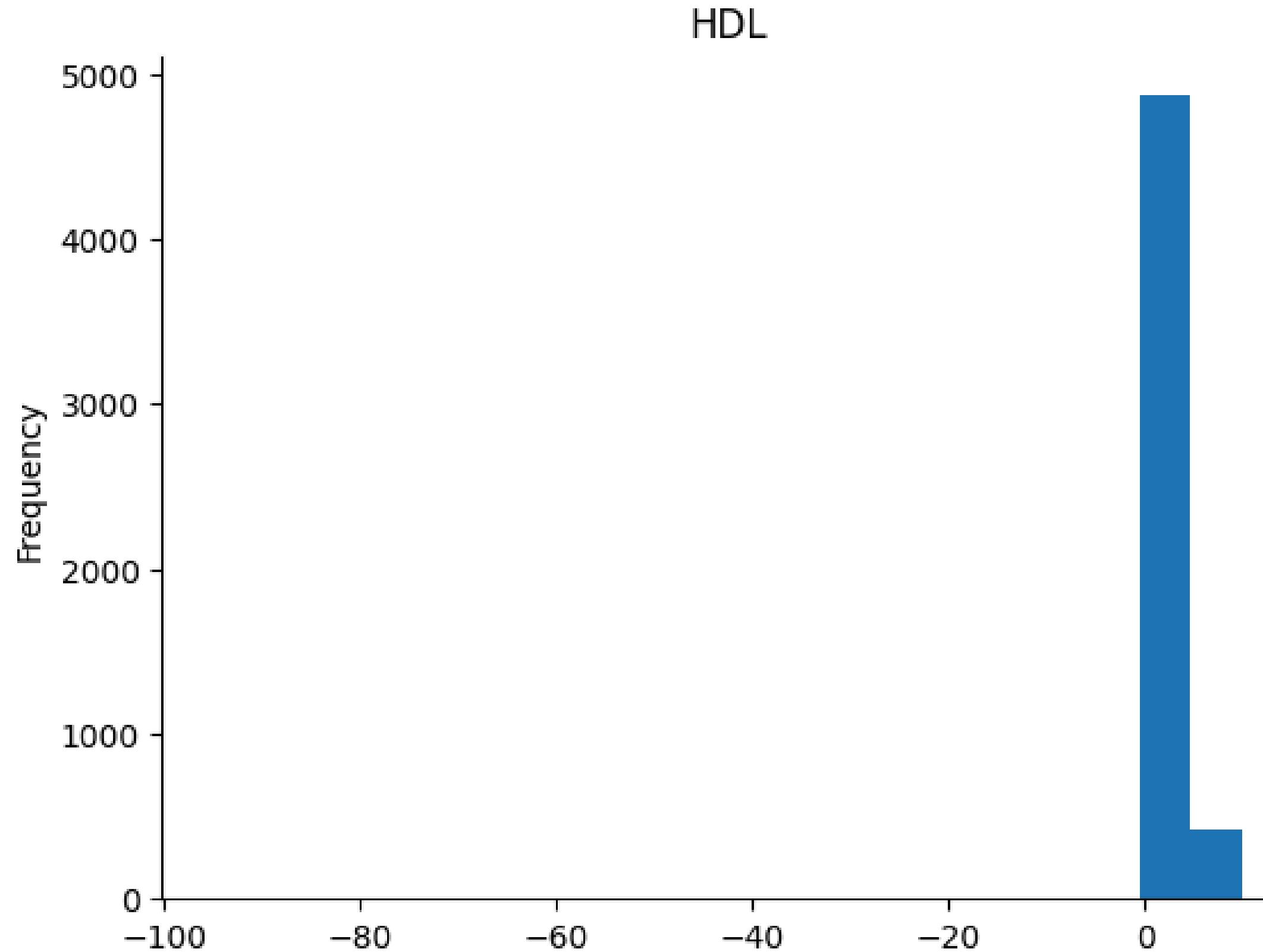


5310 non-null

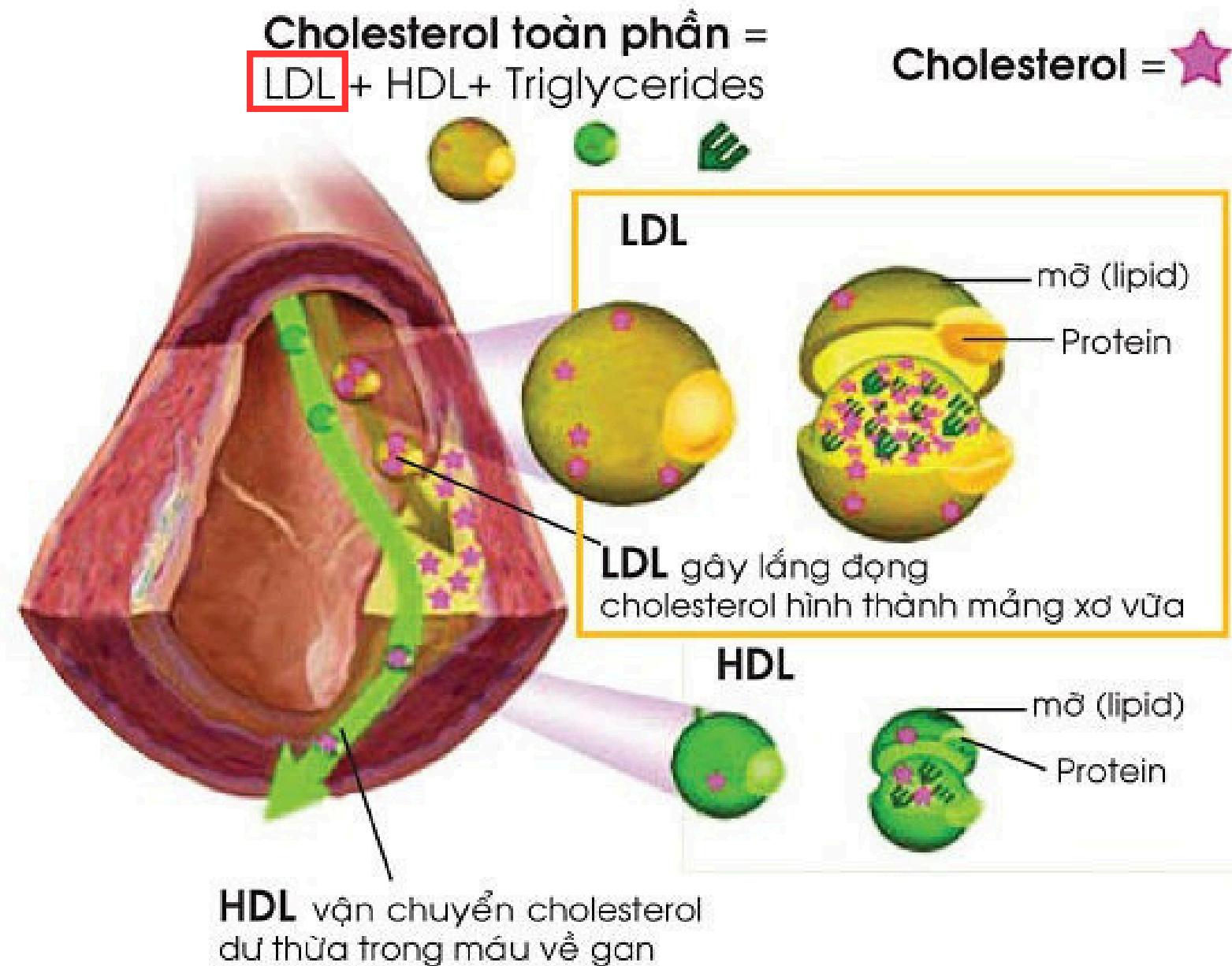
float64 →

Kiểu dữ liệu
(datatype)

Thuộc tính ‘HDL’



Thuộc tính ‘LDL’



CÁC THÀNH PHẦN MỠ MÁU

LDL (Low-Density Lipoprotein) là chỉ số Lipoprotein tỷ trọng thấp.

Lipoprotein tỷ trọng thấp các cholesterol có hại làm tăng nguy cơ xơ vữa động mạch.

Thuộc tính ‘LDL’

mean	2.705066	Chỉ số LDL trung bình
std	3.910821	Độ lệch chuẩn
min	-98.000000	Chỉ số LDL thấp nhất
25%	2.270000	Tứ phân vị Q1
50%	2.780000	Giá trị trung vị (mean)
75%	3.390000	Tứ phân vị Q3
max	9.900000	Chỉ số LDL cao nhất

Số lượng giá trị
không null

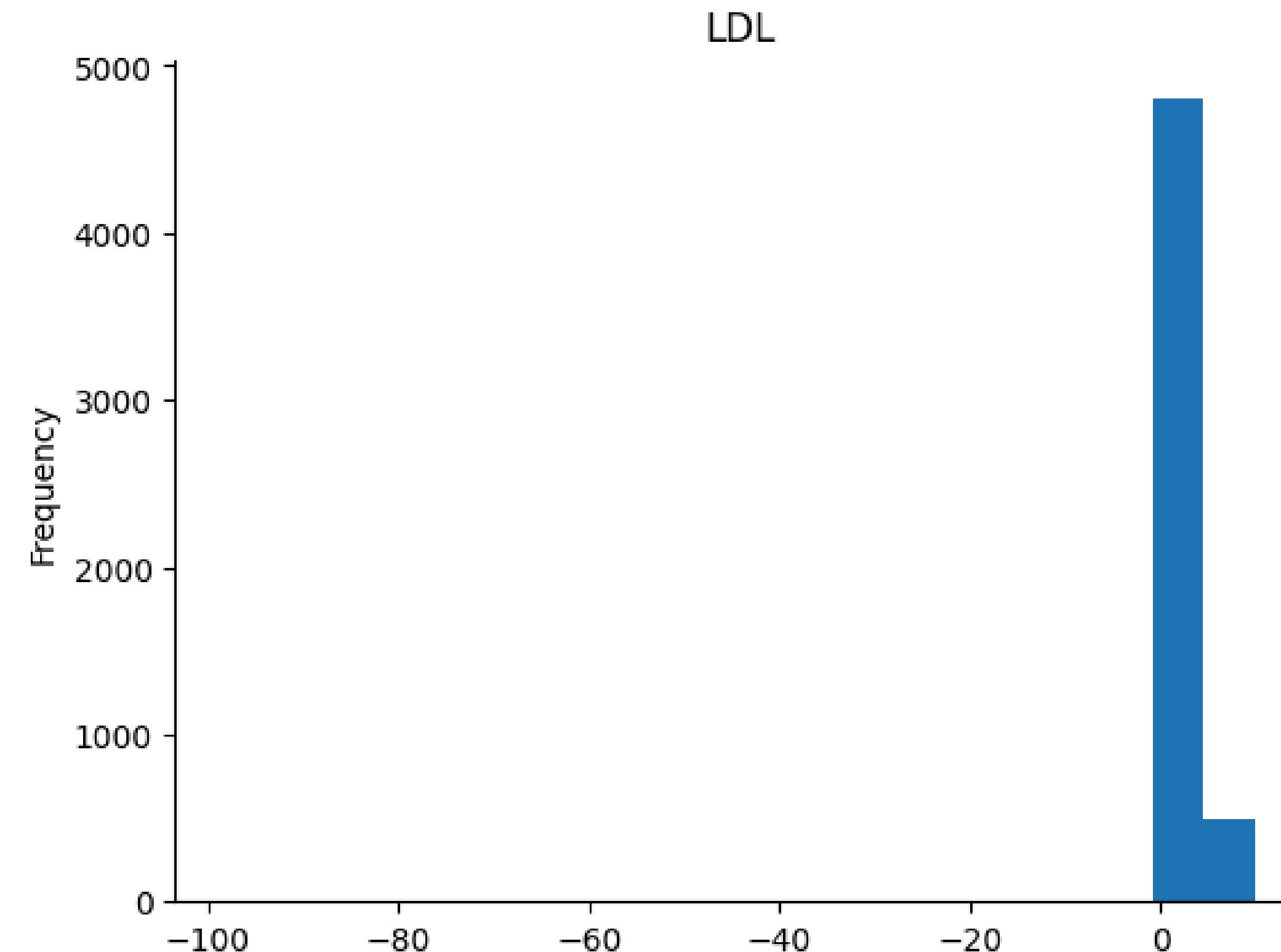


5314 non-null

float64 →

Kiểu dữ liệu
(datatype)

Thuộc tính ‘LDL’

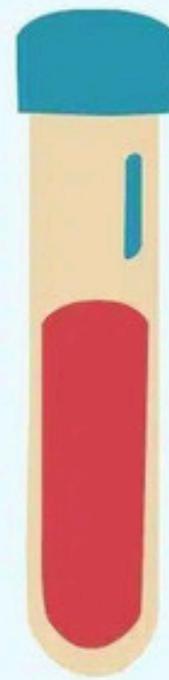


Thuộc tính 'Cr'



Endo Clinic

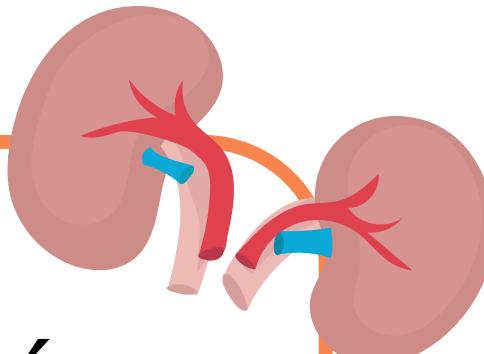
CHỈ SỐ CREATININ MÁU BÌNH THƯỜNG LÀ BAO NHIÊU?



Bình thường:

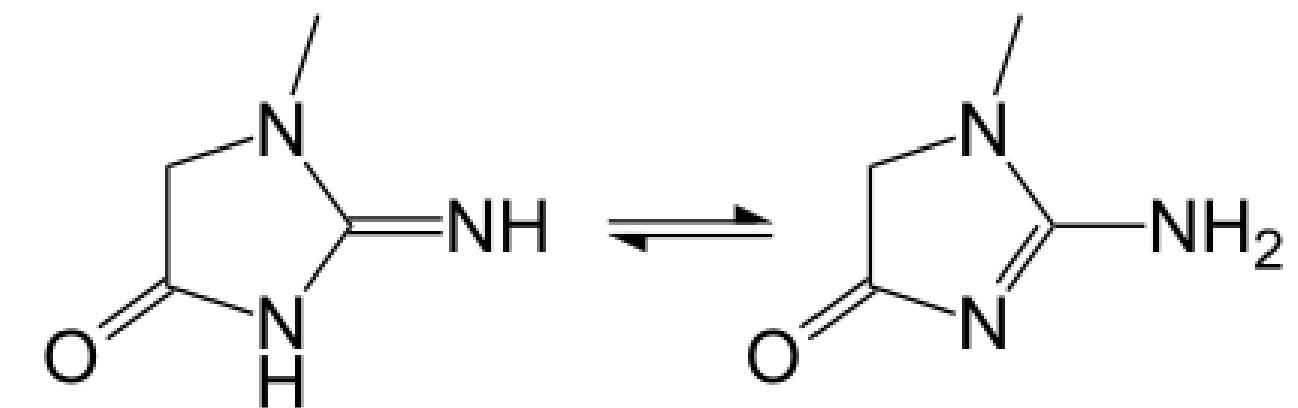
Nam là: 0,7 - 1,3 mg/dL (62 - 115 µmol/L)

Nữ là: 0,5 - 1,0 mg/dL (44 - 88 µmol/L)



Là chỉ số creatinin trong máu

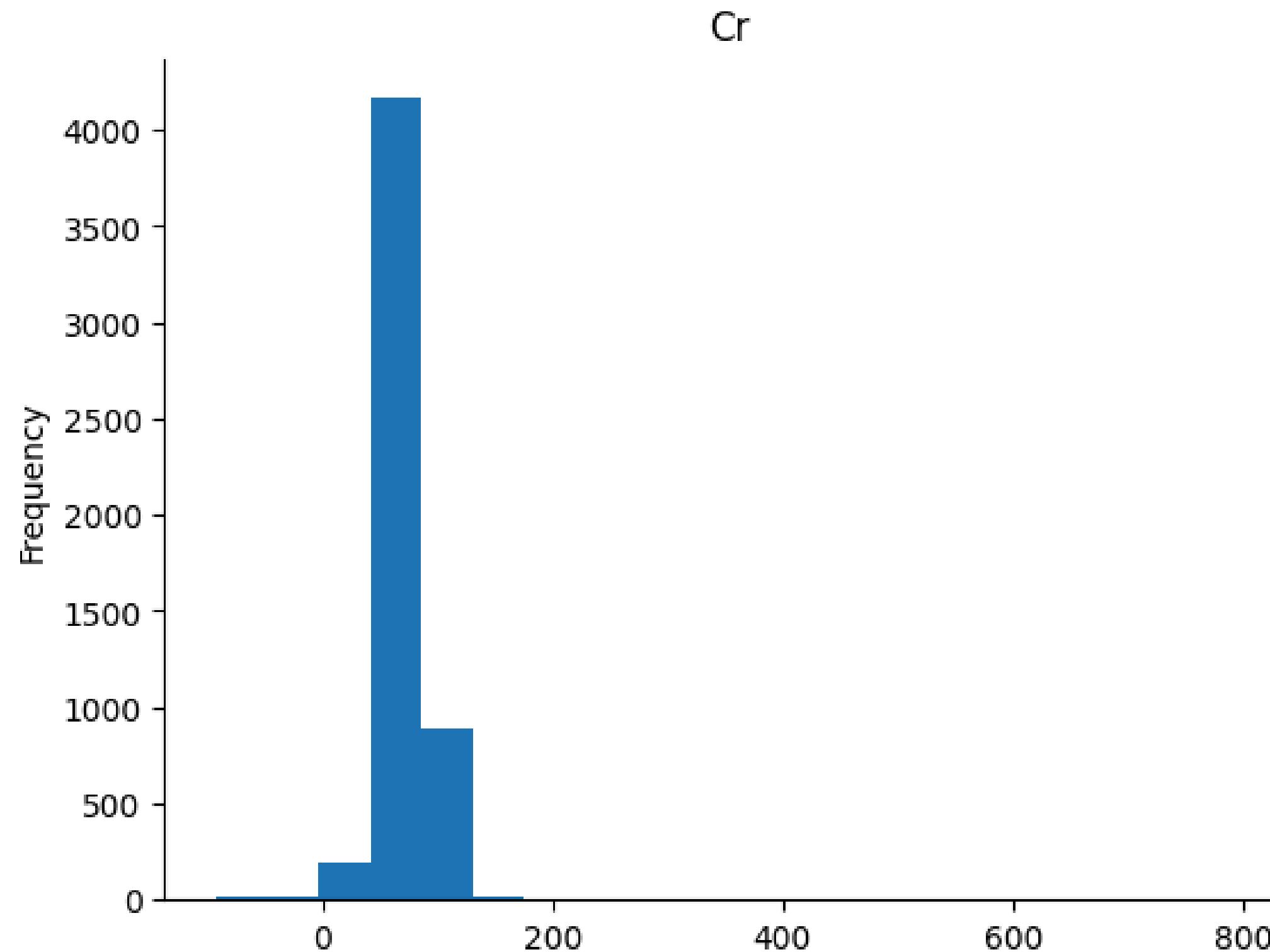
Creatinin là một chất cặn bã được đào thải thông qua thận, từ đó phản ánh chức năng thận



Thuộc tính ‘Cr’

mean	70.626207	Chỉ số creatinin trung bình
std	29.763911	Độ lệch chuẩn
min	-93.000000	Chỉ số creatinin thấp nhất
25%	57.750000	Tứ phân vị Q1
50%	70.000000	Giá trị trung vị (mean)
75%	81.400000	Tứ phân vị Q3
max	800.000000	Chỉ số creatinin cao nhất
Số lượng giá trị không null	← 5311 non-null →	Kiểu dữ liệu (datatype)

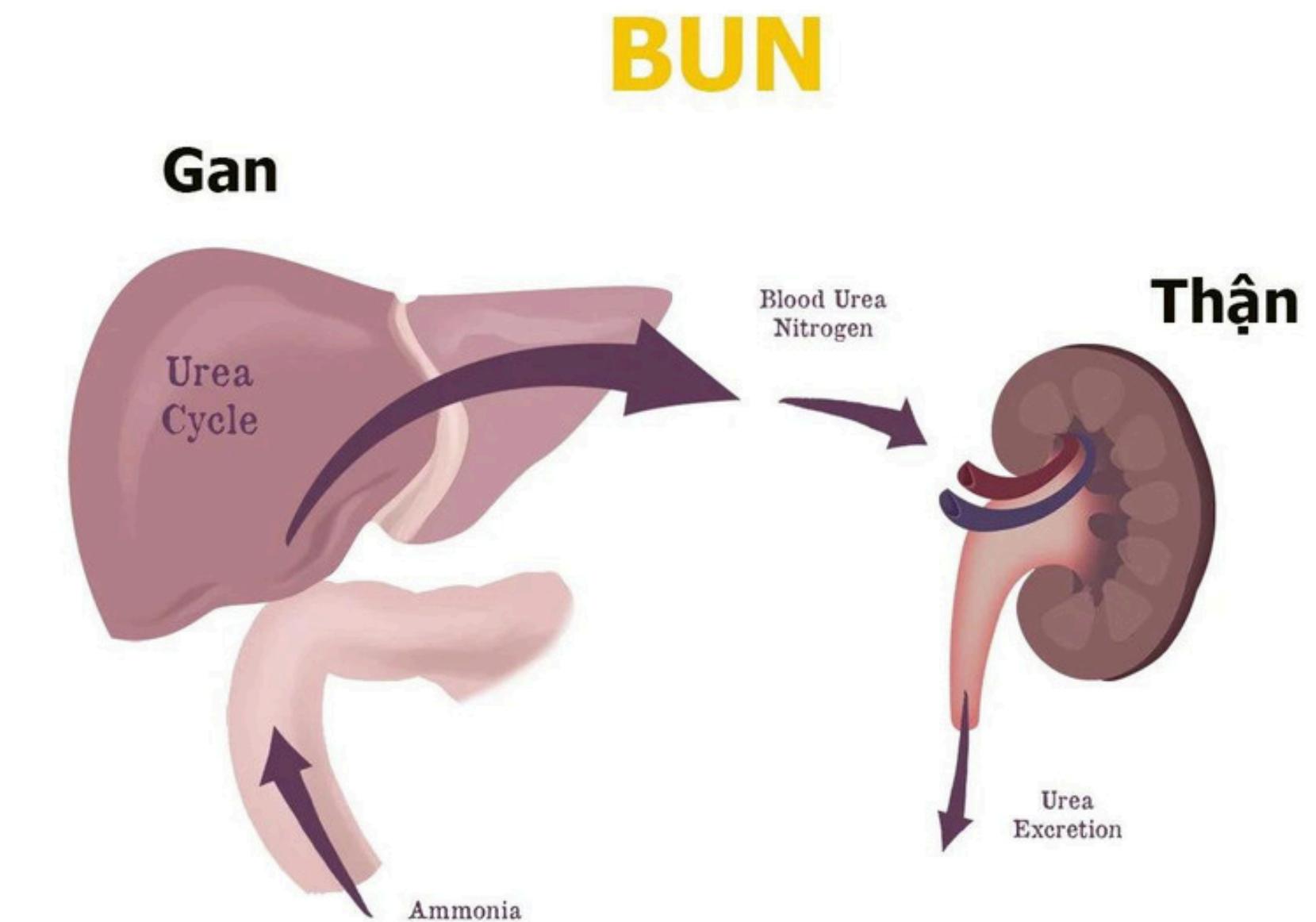
Thuộc tính ‘Cr’



Thuộc tính 'BUN'

BUN (Blood Urea Nitrogen) là chỉ số BUN dùng để đánh giá chức năng gan và thận

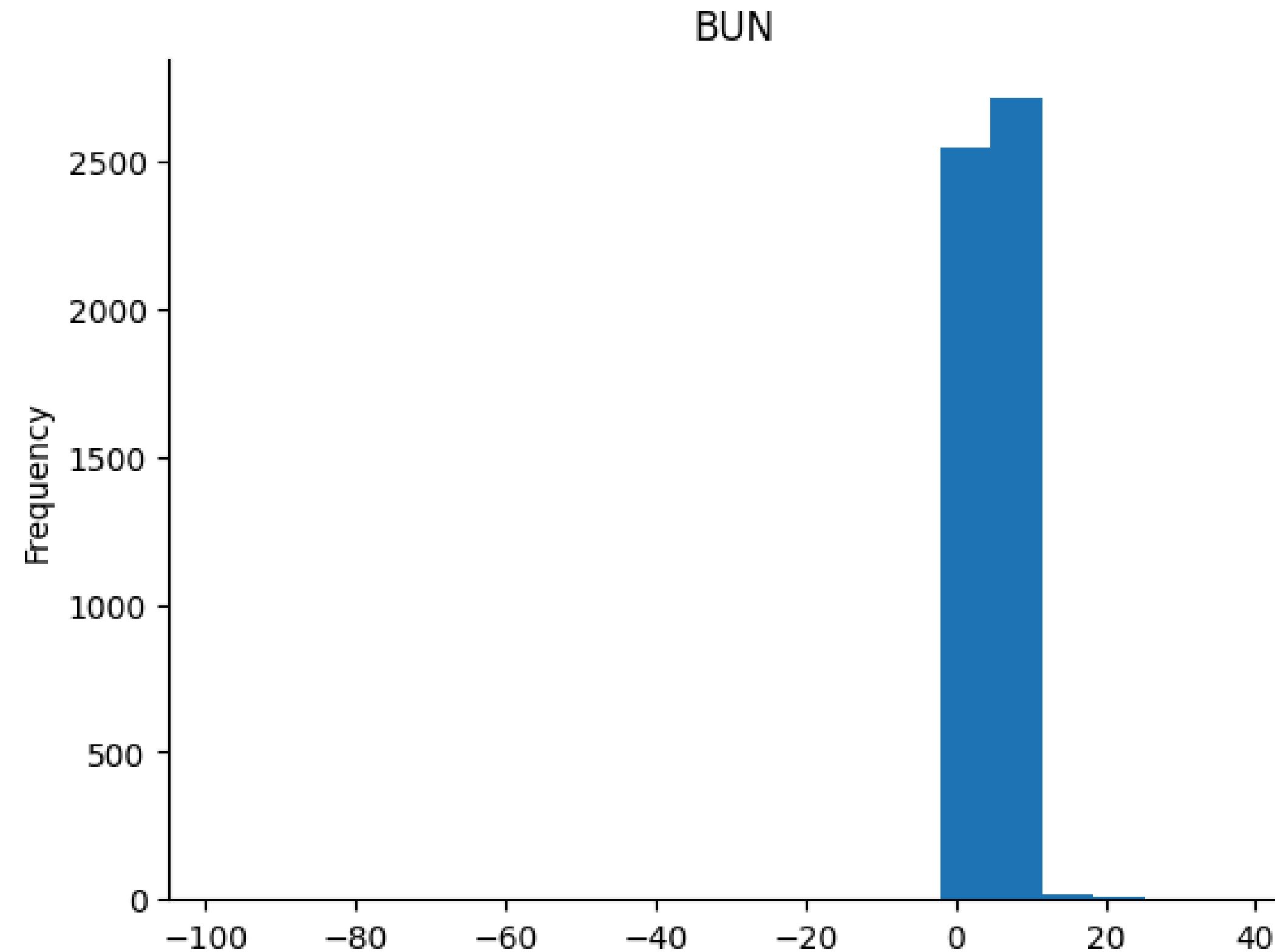
Xét nghiệm BUN sẽ cho biết nồng độ urea nitrogen trong máu đang ở mức bình thường hay bất thường



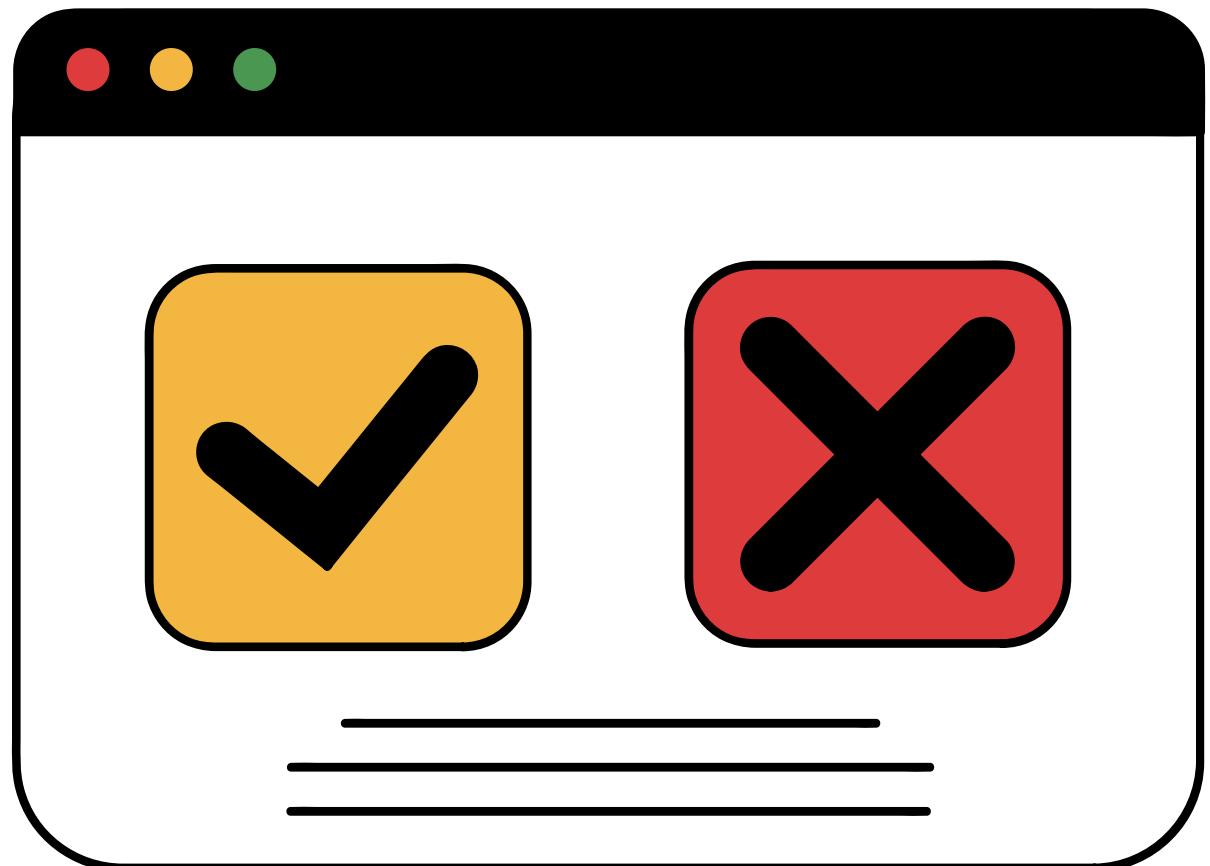
Thuộc tính ‘BUN’

mean	4.596416	Chỉ số BUN trung bình
std	4.919204	Độ lệch chuẩn
min	-98.000000	Chỉ số BUN thấp nhất
25%	3.900000	Tứ phân vị Q1
50%	4.710000	Giá trị trung vị (mean)
75%	5.600000	Tứ phân vị Q3
max	38.900000	Chỉ số BUN cao nhất
Số lượng giá trị không null	← 5311 non-null →	Kiểu dữ liệu (datatype)

Thuộc tính ‘BUN’



Thuộc tính ‘Diagnosis’



Là lớp của từng bệnh nhân bao gồm 2 nhãn 0 và 1. Bệnh nhân được chẩn đoán mắc bệnh tiểu đường được gán nhãn là 1 và ngược lại có nhãn là 0.

Thuộc tính ‘Diagnosis’

mean	0.203087	Giá trị trung bình
std	3.366983	Độ lệch chuẩn
min	-83.000000	Gía trị thấp nhất
25%	0.000000	Tứ phân vị Q1
50%	0.000000	Giá trị trung vị (mean)
75%	1.000000	Tứ phân vị Q3
max	1.000000	Gia trị cao nhất

Số lượng giá trị
không null



5313 non-null

float64



Kiểu dữ liệu
(datatype)

oooo

#	Column	Non-Null Count	Dtype
0	Age	5310 non-null	float64
1	Gender	5332 non-null	object
2	BMI	5309 non-null	float64
3	Chol	5306 non-null	float64
4	TG	5300 non-null	float64
5	HDL	5310 non-null	float64
6	LDL	5314 non-null	float64
7	Cr	5311 non-null	float64
8	BUN	5311 non-null	float64
9	Diagnosis	5313 non-null	float64

○ ○ ○ ○

	Age	BMI	Chol	TG	HDL	LDL	Cr	BUN	Diagnosis
count	5310.000	5309.000	5306.000	5300.000	5310.000	5314.000	5311.000	5311.000	5313.000
mean	48.638	24.130	4.690	1.504	1.346	2.705	70.626	4.596	0.203
std	15.253	7.719	3.448	3.984	4.349	3.911	29.764	4.919	3.367
min	-94.000	-99.000	-82.000	-94.000	-95.000	-98.000	-93.000	-98.000	-83.000
25%	36.000	22.000	4.180	0.900	1.090	2.270	57.750	3.900	0.000
50%	49.000	24.000	4.800	1.370	1.300	2.780	70.000	4.710	0.000
75%	59.000	27.000	5.460	2.100	1.590	3.390	81.400	5.600	1.000
max	93.000	47.000	11.650	32.640	9.900	9.900	800.000	38.900	1.000

o o o o



```
1 print(df.isna().sum())
```



```
Age           22  
Gender        0  
BMI           23  
Chol          26  
TG            32  
HDL           22  
LDL           18  
Cr            21  
BUN           21  
Diagnosis     19  
dtype: int64
```



```
1 duplicated_rows = df[df.duplicated()]  
2 print(len(duplicated_rows))
```



```
200
```

.....



TIỀN XỬ LÝ DỮ LIỆU

CÁC TRƯỜNG HỢP CẦN XỬ LÝ

Chuẩn hóa thuộc tính ‘Gender’ về dạng binary F và M

Loại bỏ các dòng dữ liệu có giá trị không hợp lệ
(vd: giá trị âm)

Loại bỏ các dòng dữ liệu có giá trị rỗng (NaN)

Loại bỏ các dòng dữ liệu bị trùng lặp

CHUẨN

HÓA

THUỘC

TÍNH

'GENDER'

VỀ DẠNG

BINARY

```
df['Gender'] = df['Gender'].replace('f', 'F')
```

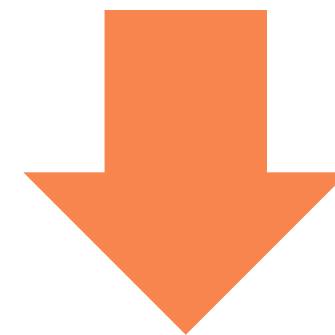
Thuộc tính 'Gender' sau khi chuẩn hóa:

Gender	Gender
M	3365
F	1966
f	1



**LOẠI BỎ
CÁC
DÒNG
DỮ LIỆU
BỊ
TRÙNG
LẶP**

```
1 df = df.drop_duplicates()  
2 df = df.reset_index()  
3 df = df.drop(df.columns[0], axis = 1)
```



```
1 duplicated_rows = df[df.duplicated()]  
2 print(len(duplicated_rows))
```

0 ✓

**LOẠI BỎ
CÁC
DÒNG
DỮ LIỆU
BỊ
TRÙNG
LẶP**

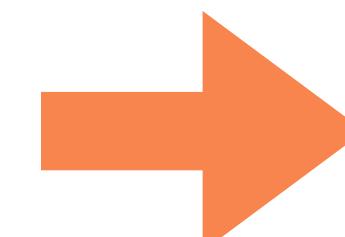
#	Column	Non-Null Count	Dtype
0	Age	5110 non-null	object
1	Gender	5132 non-null	object
2	BMI	5109 non-null	object
3	Chol	5107 non-null	object
4	TG	5101 non-null	object
5	HDL	5110 non-null	object
6	LDL	5114 non-null	object
7	Cr	5111 non-null	object
8	BUN	5112 non-null	object
9	Diagnosis	5114 non-null	object

**LOẠI BỎ
CÁC
DÒNG
DỮ LIỆU
có
GIÁ TRỊ
RỖNG**

```
1 df = df.dropna()  
2 df = df.reset_index()  
3 df = df.drop(df.columns[0], axis = 1)
```

```
1 print(df.isna().sum())
```

Số lượng giá trị NaN
ở mỗi thuộc tính sau khi xử lý:



Age	0
Gender	0
BMI	0
Chol	0
TG	0
HDL	0
LDL	0
Cr	0
BUN	0
Diagnosis	0



**LOẠI BỎ
CÁC
DÒNG
DỮ LIỆU
có
GIÁ TRỊ
RỖNG**

#	Column	Non-Null Count	Dtype
0	Age	4933 non-null	object
1	Gender	4933 non-null	object
2	BMI	4933 non-null	object
3	Chol	4933 non-null	object
4	TG	4933 non-null	object
5	HDL	4933 non-null	object
6	LDL	4933 non-null	object
7	Cr	4933 non-null	object
8	BUN	4933 non-null	object
9	Diagnosis	4933 non-null	object

LOẠI BỎ

CÁC
DÒNG
DỮ LIỆU
cÓ
GIÁ TRỊ
KHÔNG
HỢP LỆ

```
1 df = df.drop(df[(df['Age'].astype(float) < 0) |  
2                   (df['BMI'].astype(float) < 0) |  
3                   (df['Chol'].astype(float) < 0) |  
4                   (df['TG'].astype(float) < 0) |  
5                   (df['HDL'].astype(float) < 0) |  
6                   (df['LDL'].astype(float) < 0) |  
7                   (df['Cr'].astype(float) < 0) |  
8                   (df['BUN'].astype(float) < 0) |  
9                   (df['Diagnosis'].astype(float) < 0)].index)  
10 df = df.reset_index()  
11 df = df.drop(df.columns[0], axis = 1)
```

LOẠI BỎ

CÁC

DÒNG

DỮ LIỆU

có

GIÁ TRỊ

KHÔNG

HỢP LỆ

#	Column	Non-Null Count	Dtype
0	Age	4743 non-null	float64
1	Gender	4743 non-null	object
2	BMI	4743 non-null	float64
3	Chol	4743 non-null	float64
4	TG	4743 non-null	float64
5	HDL	4743 non-null	float64
6	LDL	4743 non-null	float64
7	Cr	4743 non-null	float64
8	BUN	4743 non-null	float64
9	Diagnosis	4743 non-null	float64

LOẠI BỎ

CÁC

DÒNG

DỮ LIỆU

CÓ

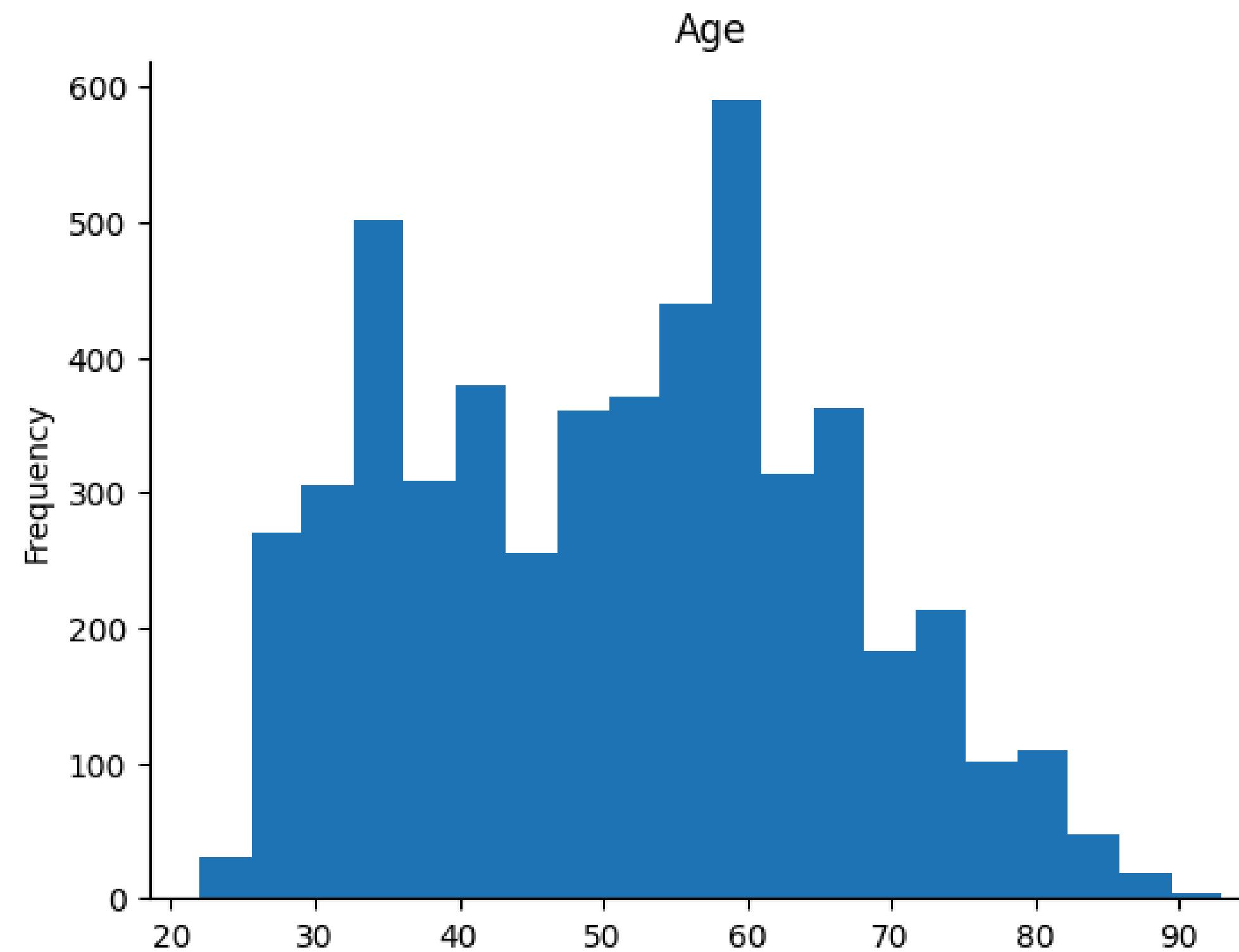
GIÁ TRỊ

KHÔNG

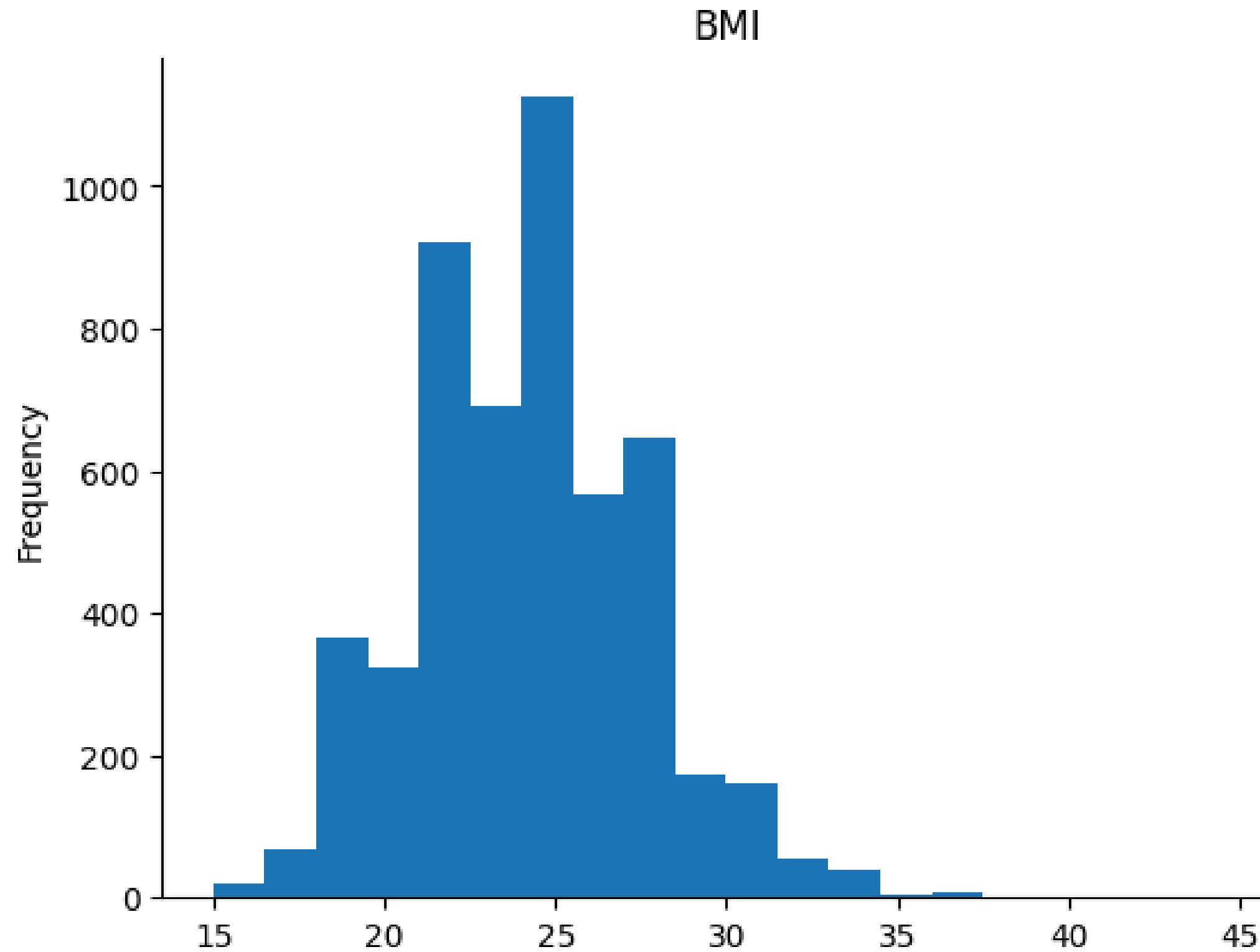
HỢP LỆ

	Age	BMI	Chol	TG	HDL	LDL	Cr	BUN	Diagnosis
count	4743.000	4743.000	4743.000	4743.000	4743.000	4743.000	4743.000	4743.000	4743.000
mean	48.980	24.613	4.875	1.723	1.595	2.917	71.201	4.892	0.390
std	14.035	4.284	1.006	1.337	1.040	0.950	29.124	1.697	0.488
min	20.000	15.000	0.000	0.000	0.000	0.300	4.861	0.500	0.000
25%	36.000	22.000	4.200	0.910	1.100	2.290	58.000	3.910	0.000
50%	49.000	24.000	4.800	1.380	1.300	2.800	70.200	4.730	0.000
75%	59.000	27.000	5.470	2.100	1.590	3.400	81.700	5.580	1.000
max	93.000	47.000	11.650	32.640	9.900	9.900	800.000	38.900	1.000

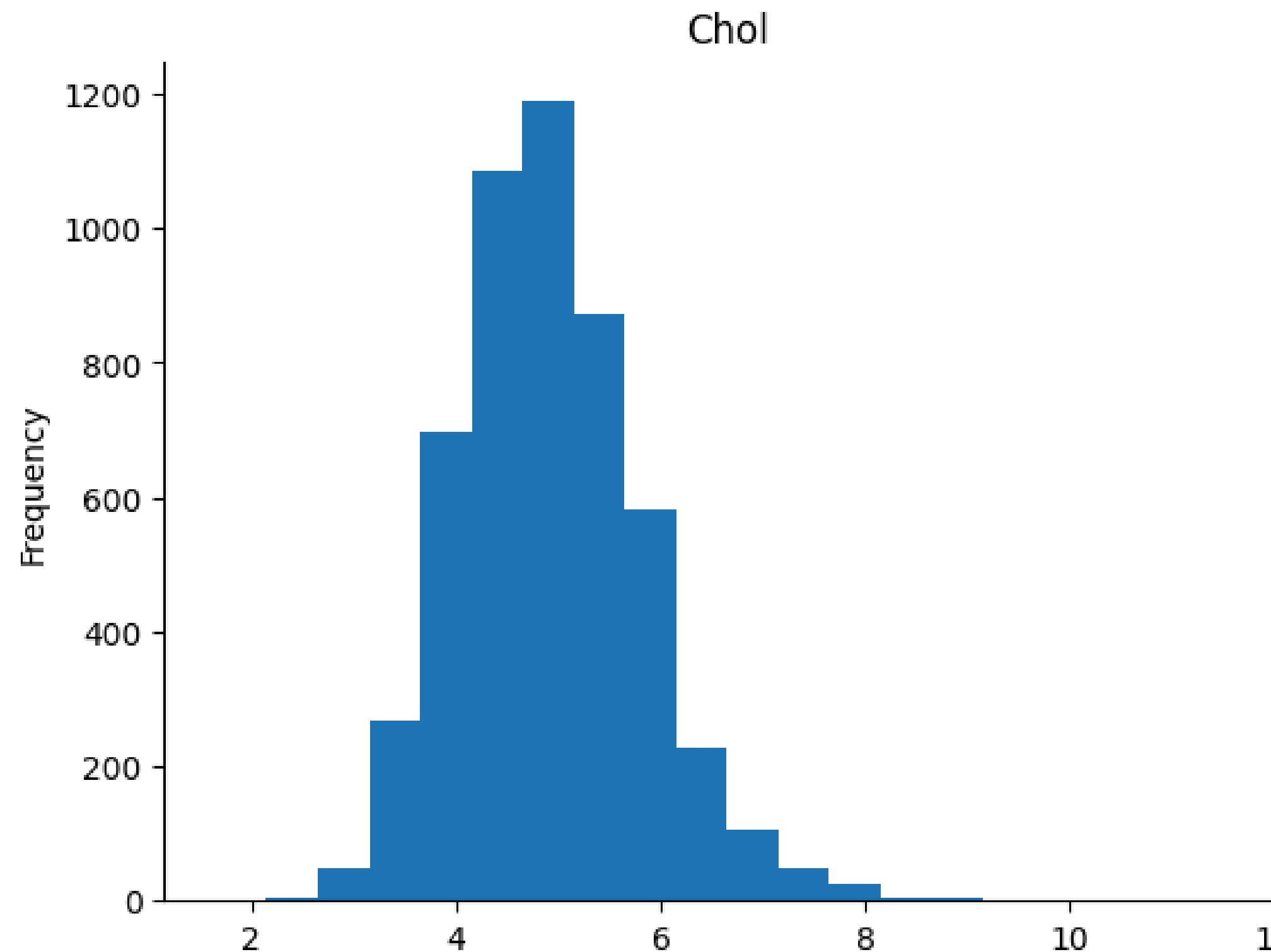
Thuộc tính ‘Age’



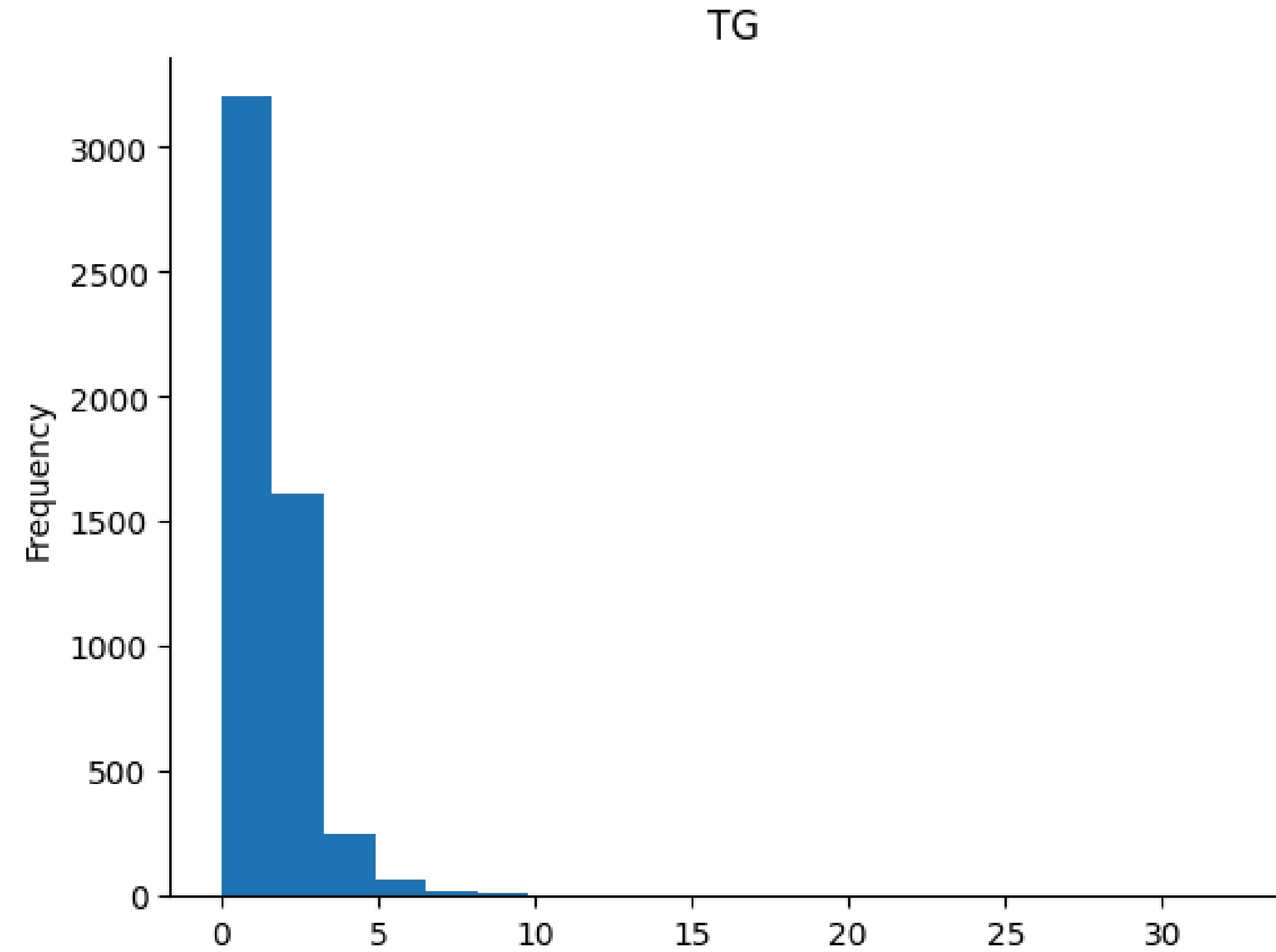
Thuộc tính ‘BMI’



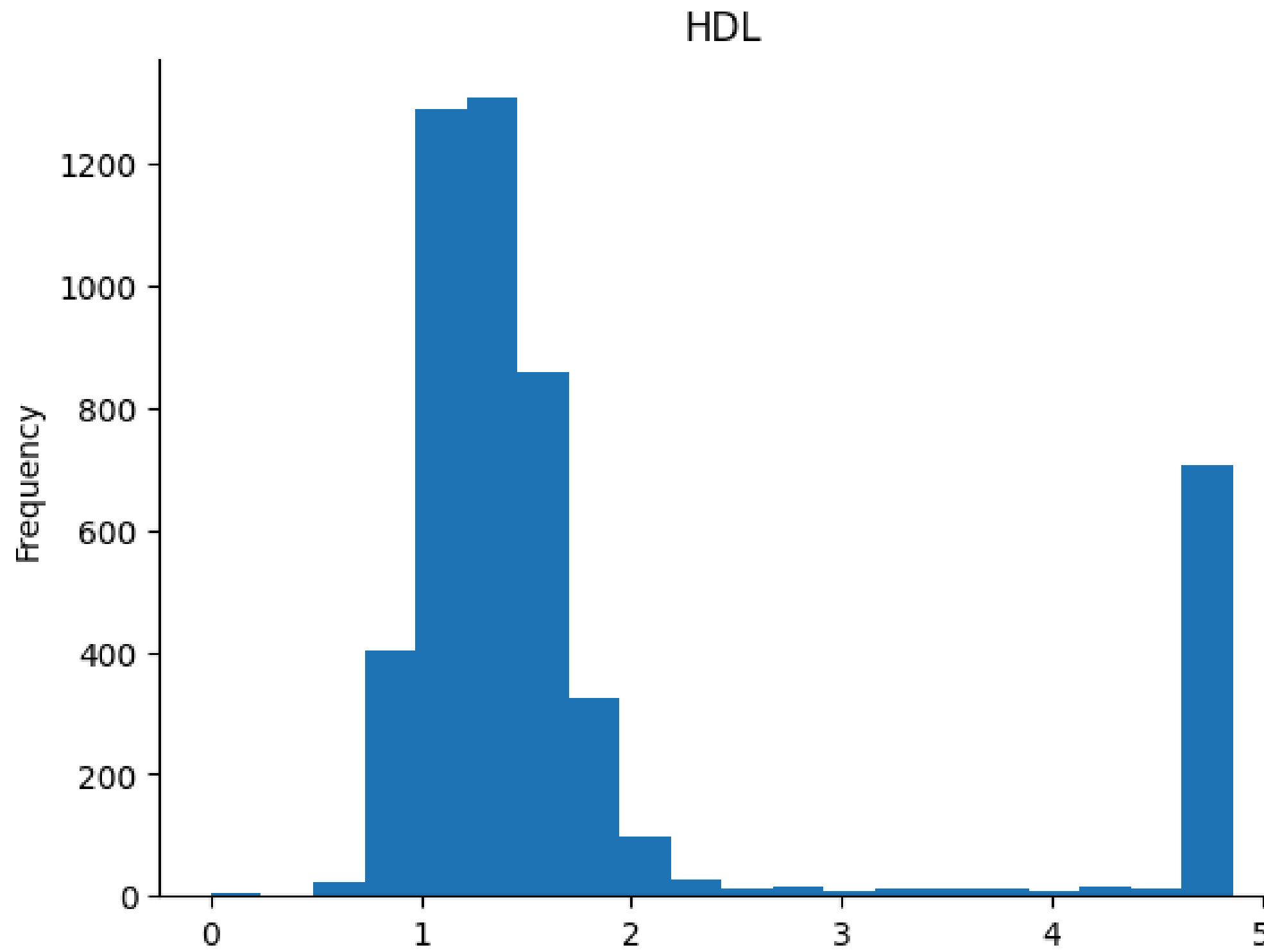
Thuộc tính ‘Chol’



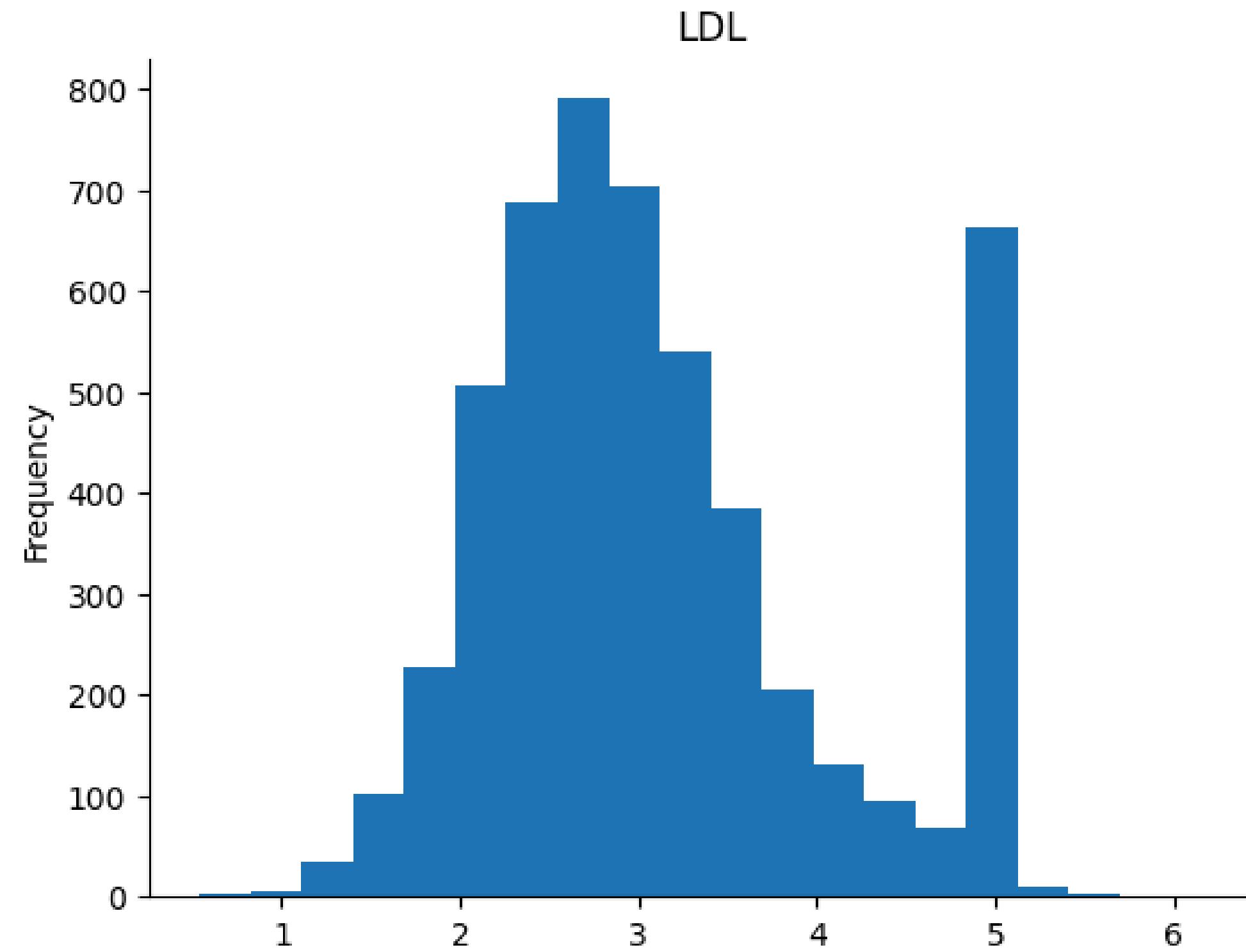
Thuộc tính 'TG'



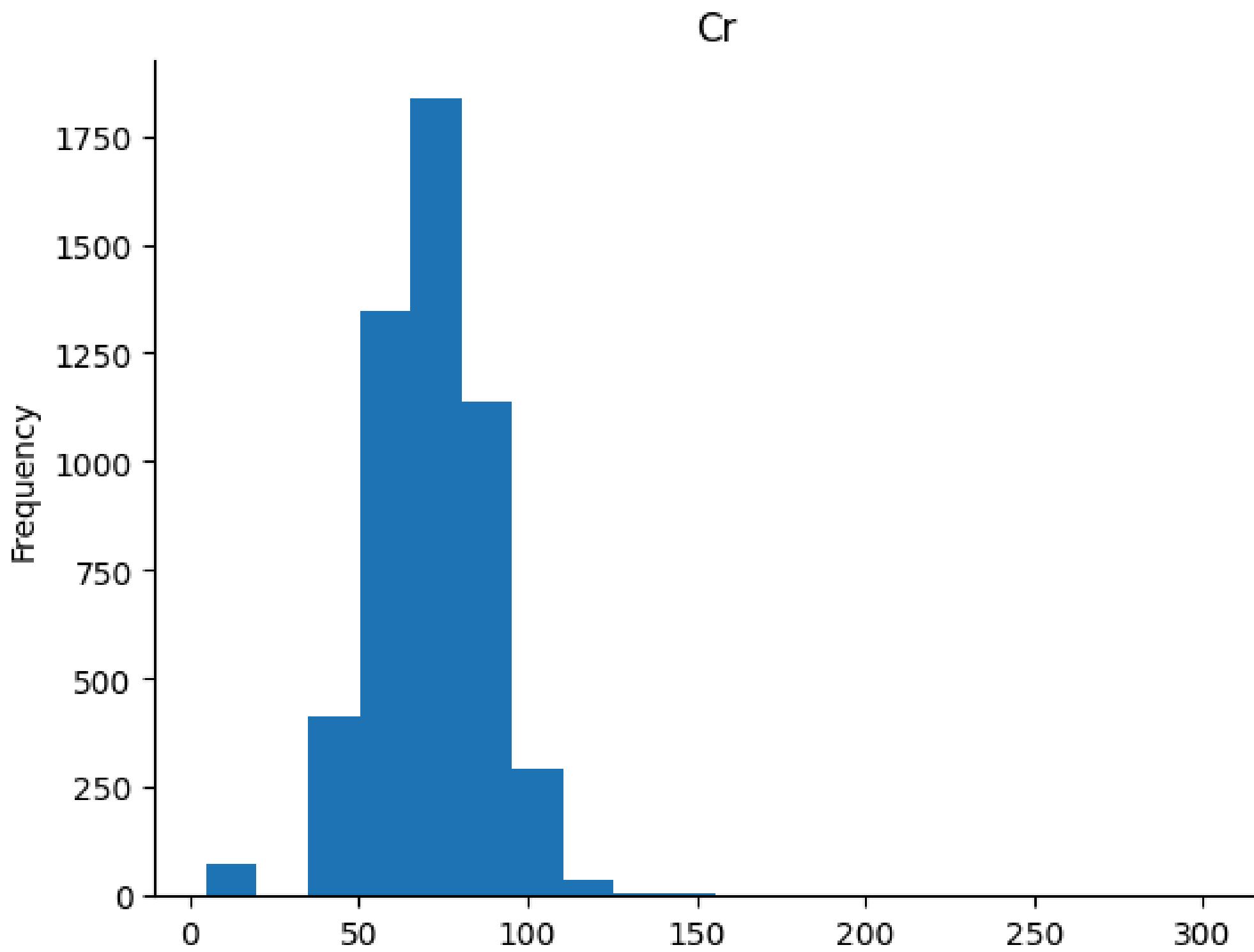
Thuộc tính ‘HDL’



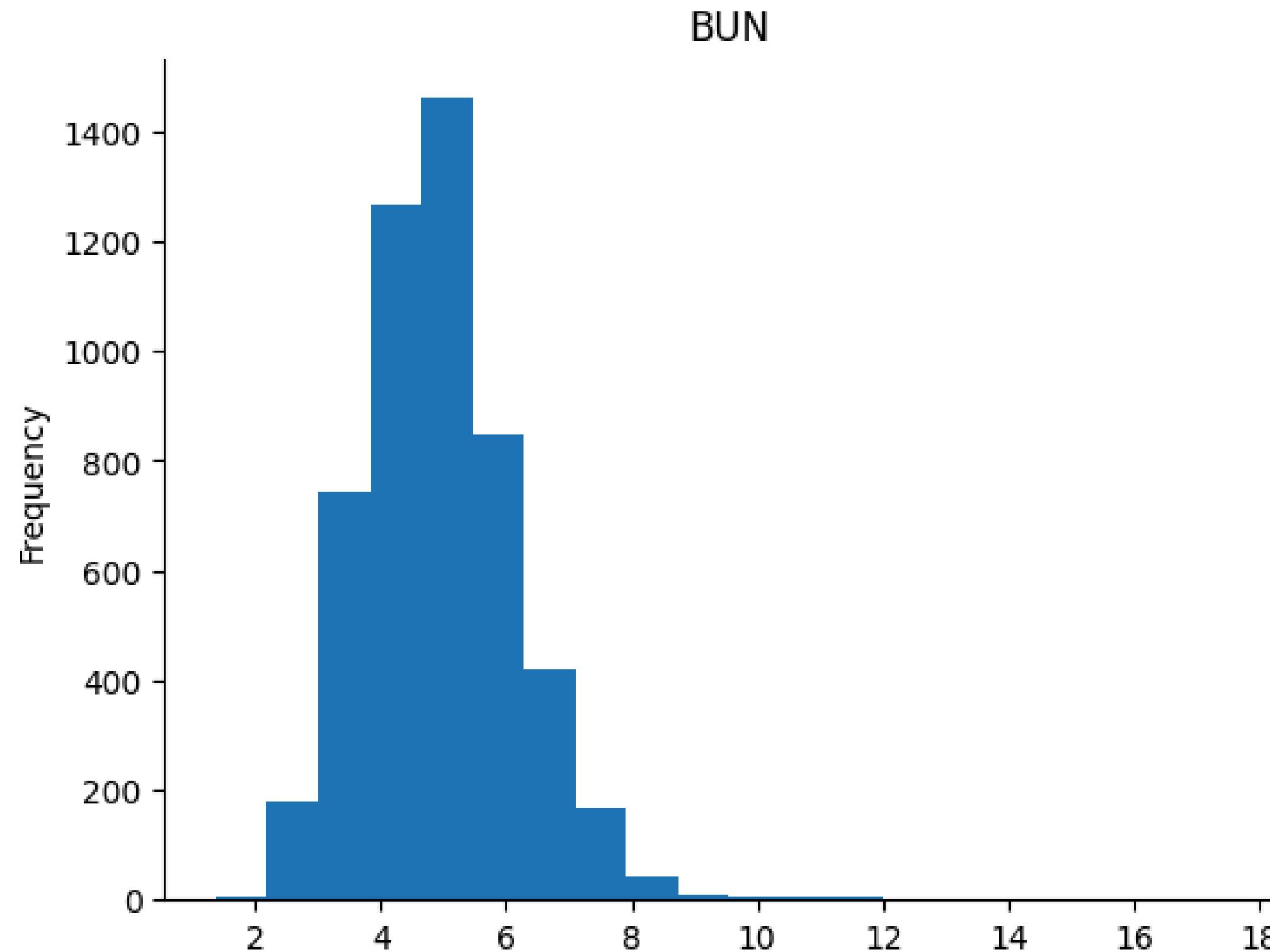
Thuộc tính ‘LDL’



Thuộc tính 'Cr'

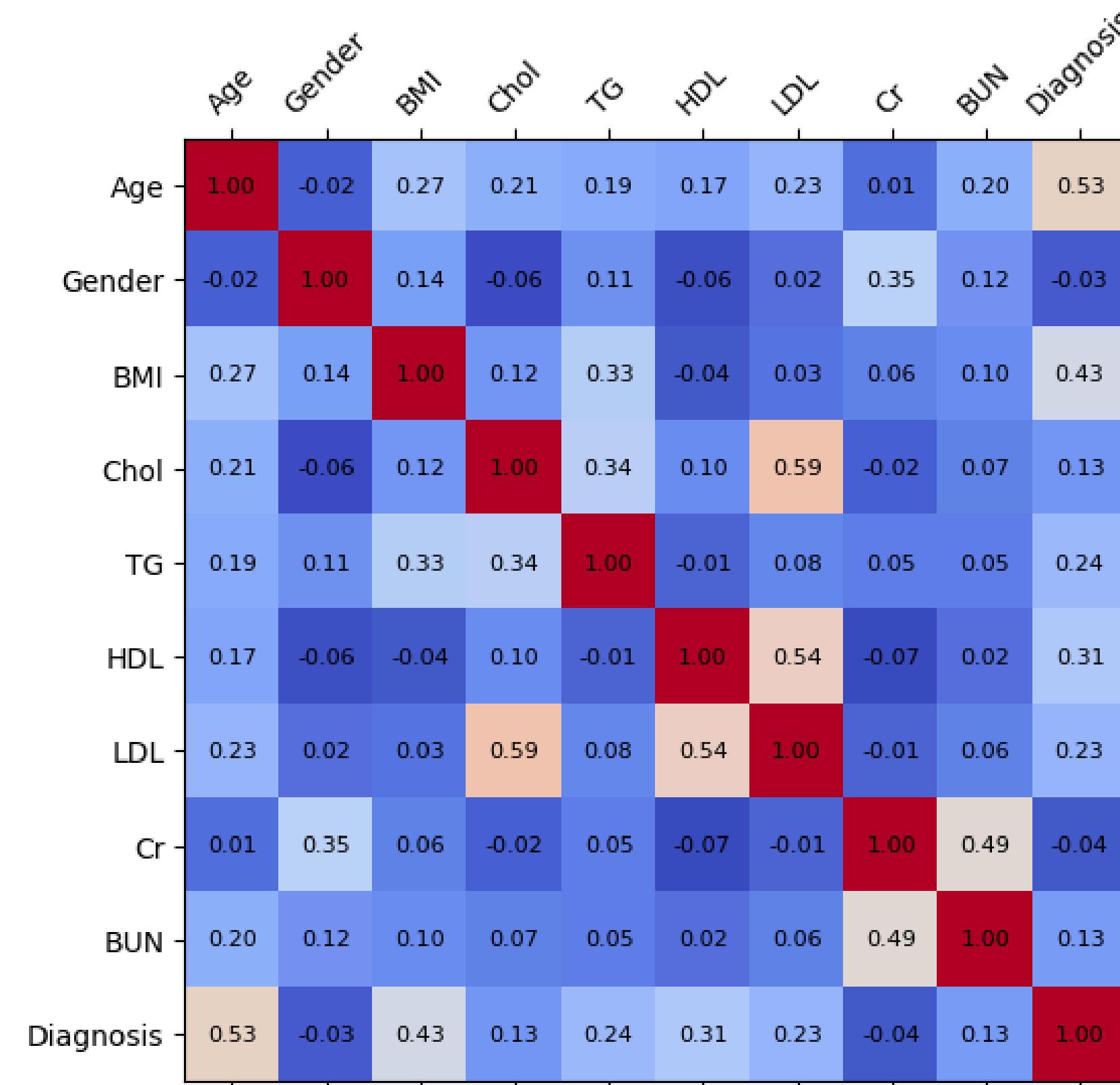


Thuộc tính ‘BUN’



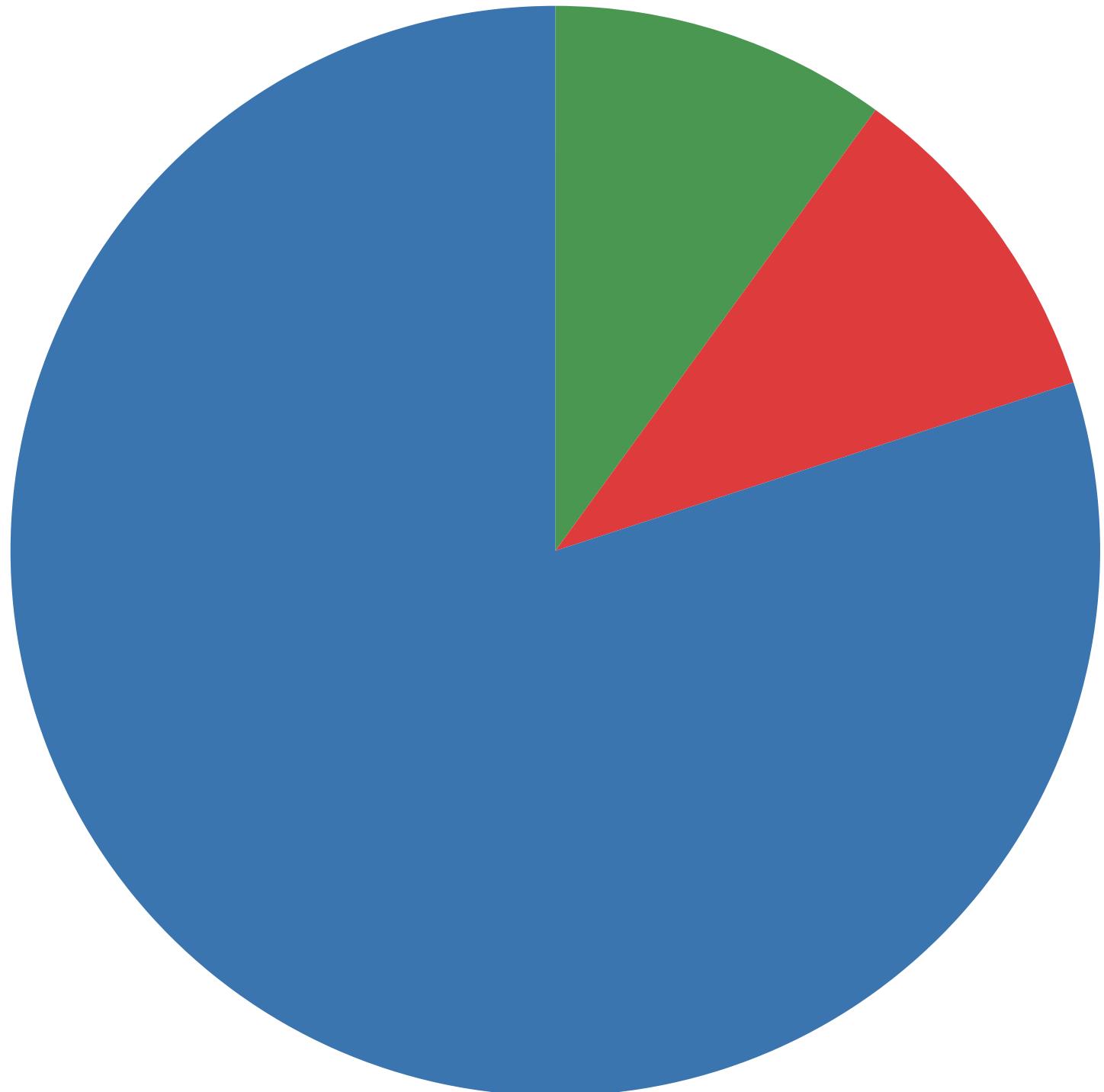
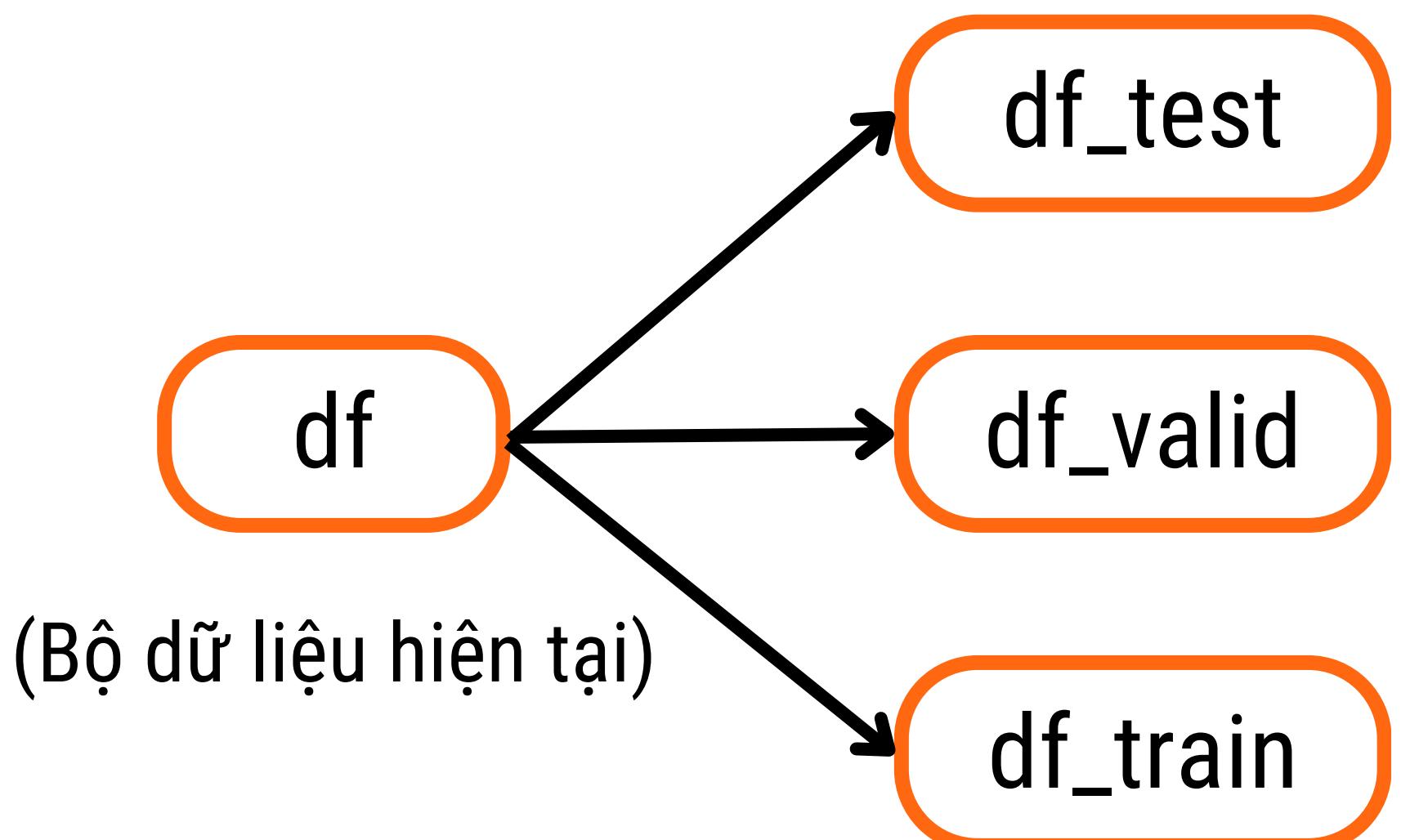
Độ tương quan giữa các thuộc tính

Correlation Heatmap



Chia bộ dữ liệu hiện tại thành 3

df_test df_valid df_train



KIỂM TRA SỰ CÂN BẰNG CỦA DỮ LIỆU

Diagnosis	
0.0	2583
1.0	1211



- => Số lượng giá trị 0 nhiều hơn giá trị 1 xấp xỉ 2,13 lần
- => Giữa các giá trị có sự chênh lệch cao
- => Dữ liệu không cân bằng

oooo

+++++
+++++
++
++
++

SMOTE

Synthetic Minority Over-sampling Technique là kỹ thuật xử lý mất cân bằng dữ liệu bằng cách tăng số lượng thể hiện của class thiểu số mà không thay đổi số lượng thể hiện của class đa số.



XỬ LÝ

VĂN ĐỀ

DỮ LIỆU

KHÔNG

CÂN

BẰNG

Chuẩn hóa thuộc tính Gender về giá trị nhị phân 0 và 1:

```
df_test['Gender'] = df_test['Gender'].replace('F', 0)
df_test['Gender'] = df_test['Gender'].replace('M', 1)
```

```
df_valid['Gender'] = df_valid['Gender'].replace('F', 0)
df_valid['Gender'] = df_valid['Gender'].replace('M', 1)
```

```
df_train['Gender'] = df_train['Gender'].replace('F', 0)
df_train['Gender'] = df_train['Gender'].replace('M', 1)
```

Sử dụng hàm SMOTE từ thư viện imblearn để over sampling:

```
from imblearn.over_sampling import SMOTE
```

```
smote = SMOTE(k_neighbors=5)
```

```
df_train, label_train = smote.fit_resample(df_train, label_train)
```

XỬ LÝ

VĂN ĐỀ

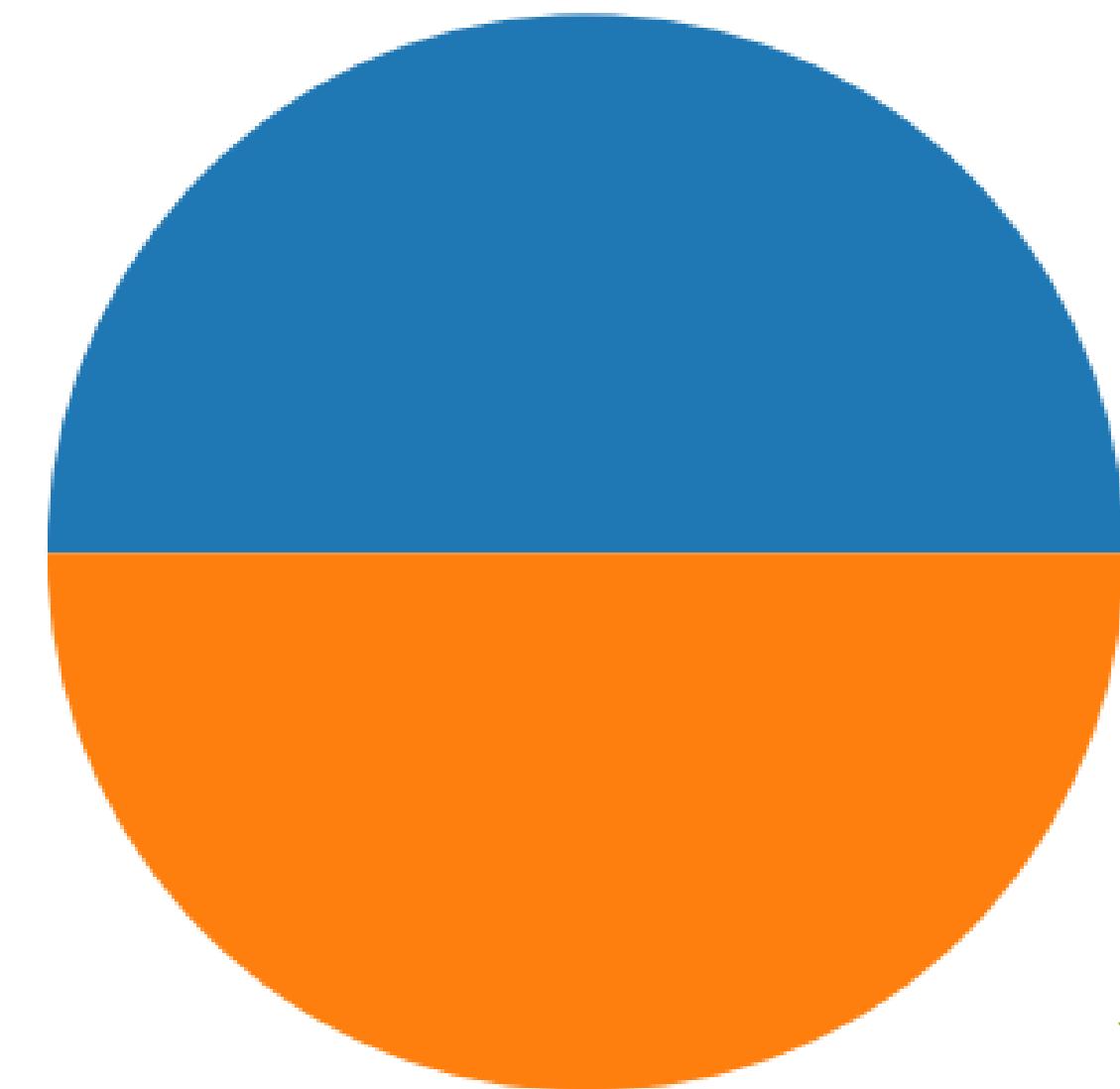
DỮ LIỆU

KHÔNG

CÂN

BẰNG

Diagnosis	
0.0	2583
1.0	2583



GOM CUM

- K-means
- Hierarchical



○ ○ ○ ○

++ + + +
+ + + + +
+ +
+ +
+ +

K-MEANS

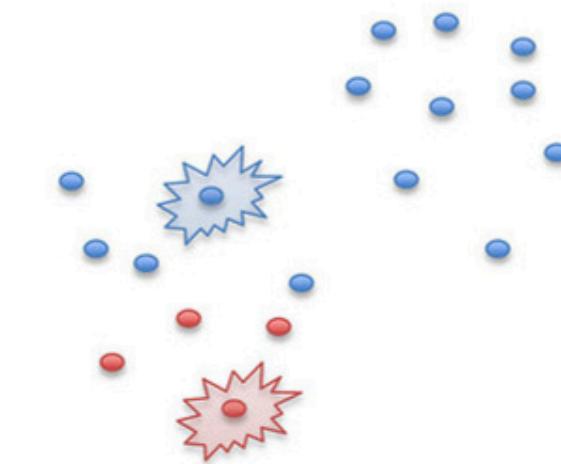
K-means là thuật toán phân cụm phổ biến, hoạt động bằng cách gán mỗi điểm dữ liệu vào cụm có tâm điểm gần nó nhất. Sau đó, nó cập nhật tâm điểm của các cụm dựa trên các điểm dữ liệu đã được gán. Quá trình này được lặp lại cho đến khi các cụm ổn định.



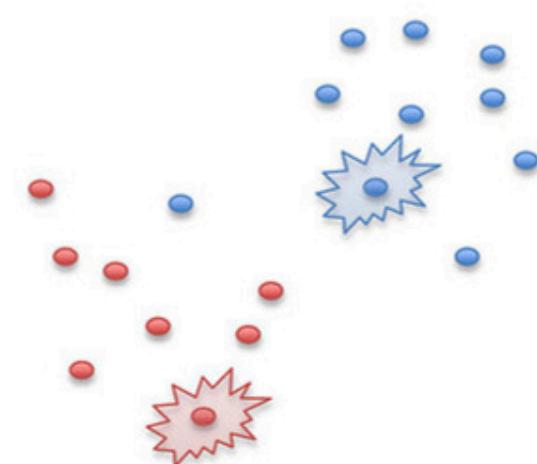
K-MEANS

- Yêu cầu số lượng cụm trước
- Nhạy cảm với điểm ngoại lệ
- Phụ thuộc vào điểm khởi tạo

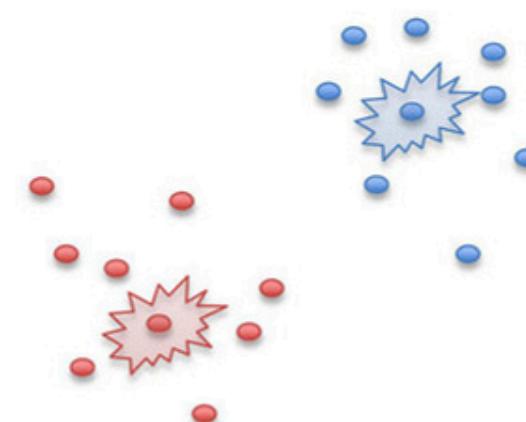
Initial Seeding



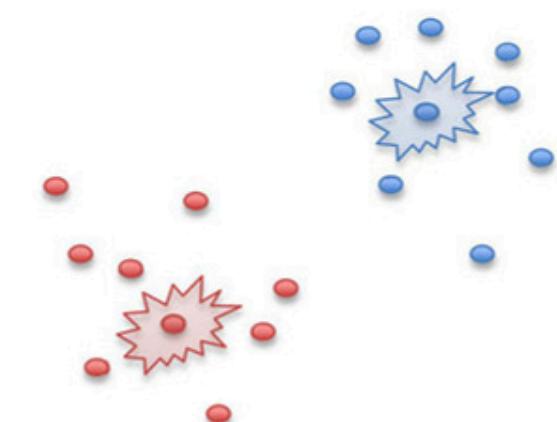
After Round 1



After Round 2

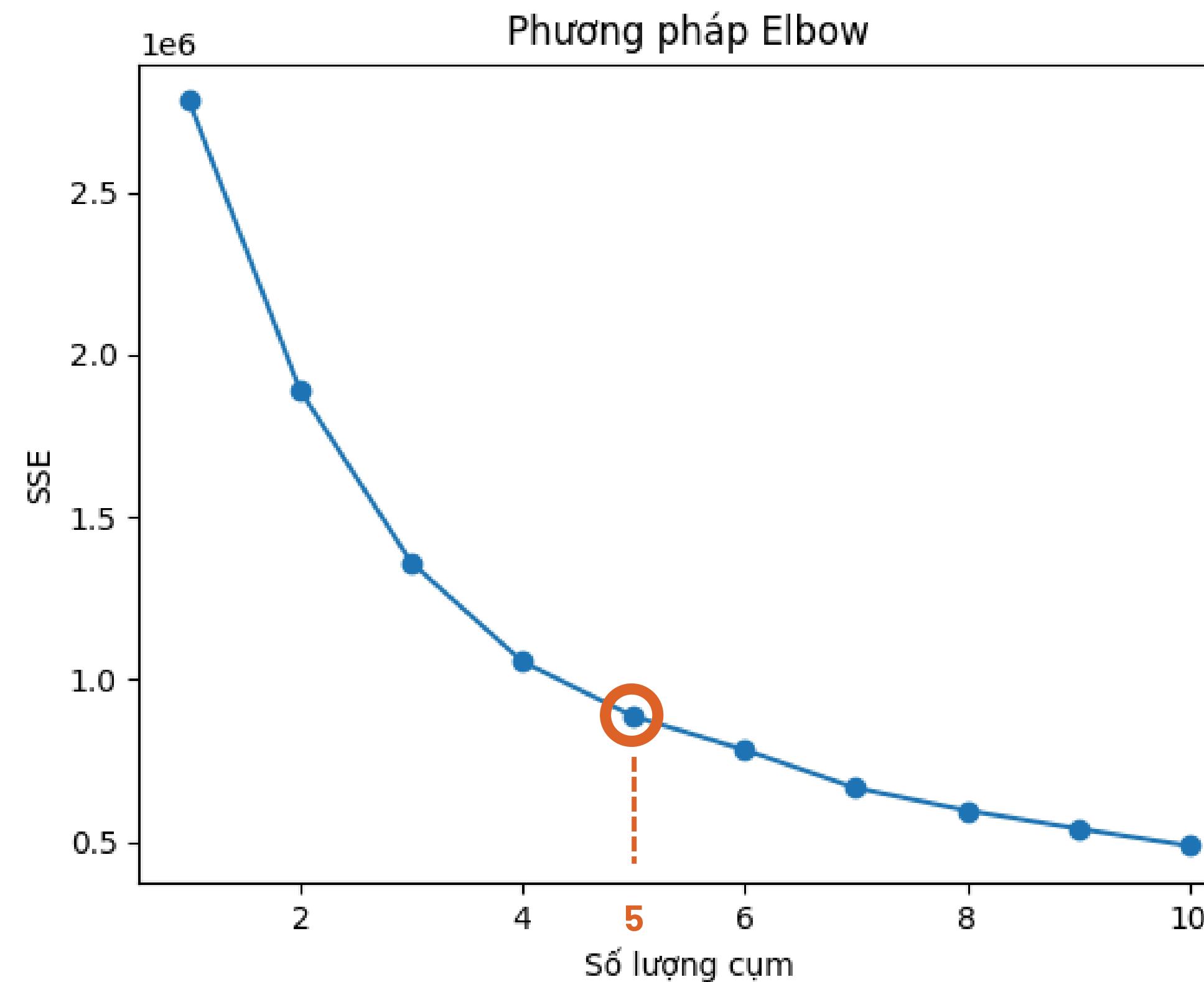


Final



++
++
++
++
++

Chọn số cụm tối ưu bằng biểu đồ Elbow



=> Chọn 5 cụm

num_clusters = 5

KẾT QUẢ

THU

ĐƯỢC

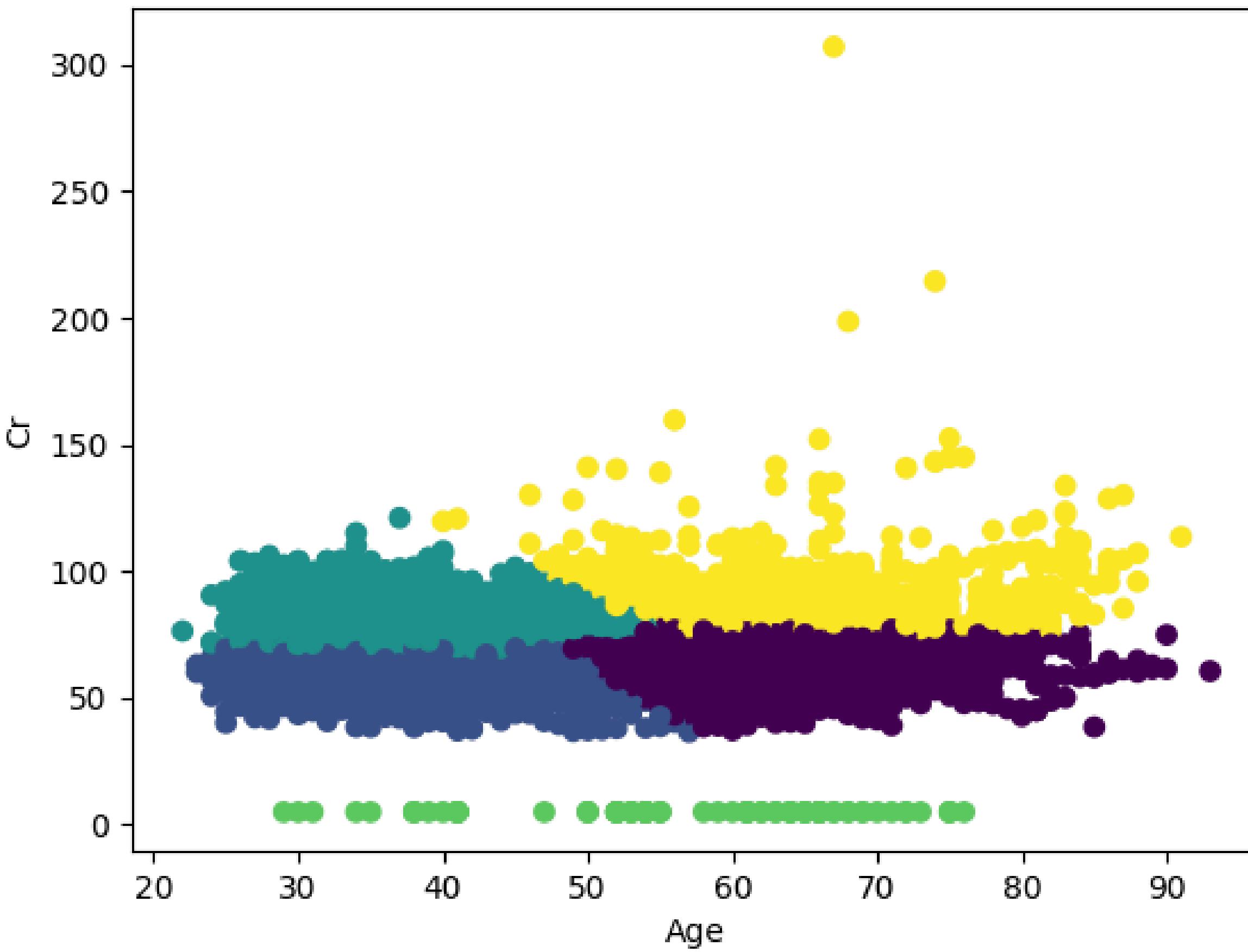
SAU KHI

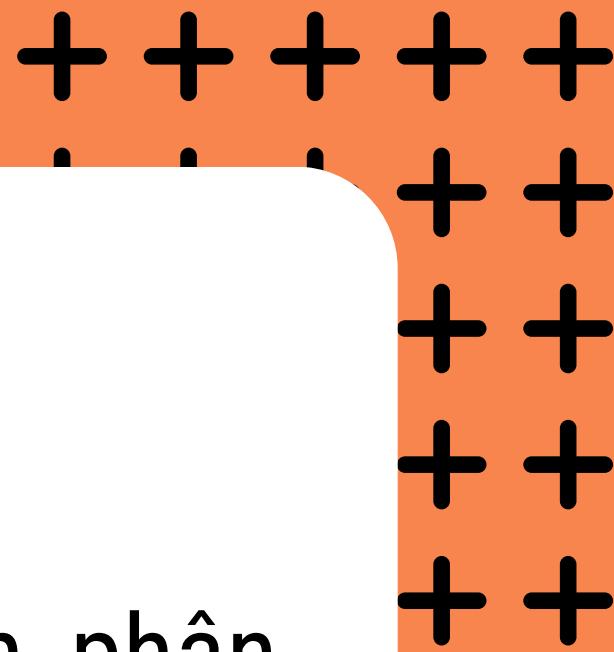
GOM CỤM

BẰNG

K-MEANS

Phân cụm K-Means





HIERARCHICAL

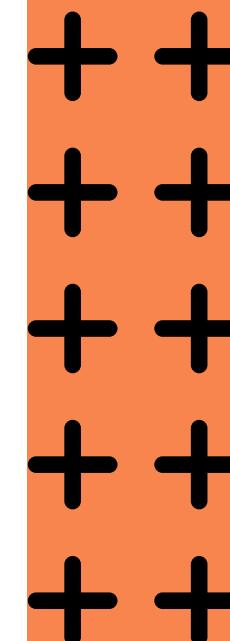
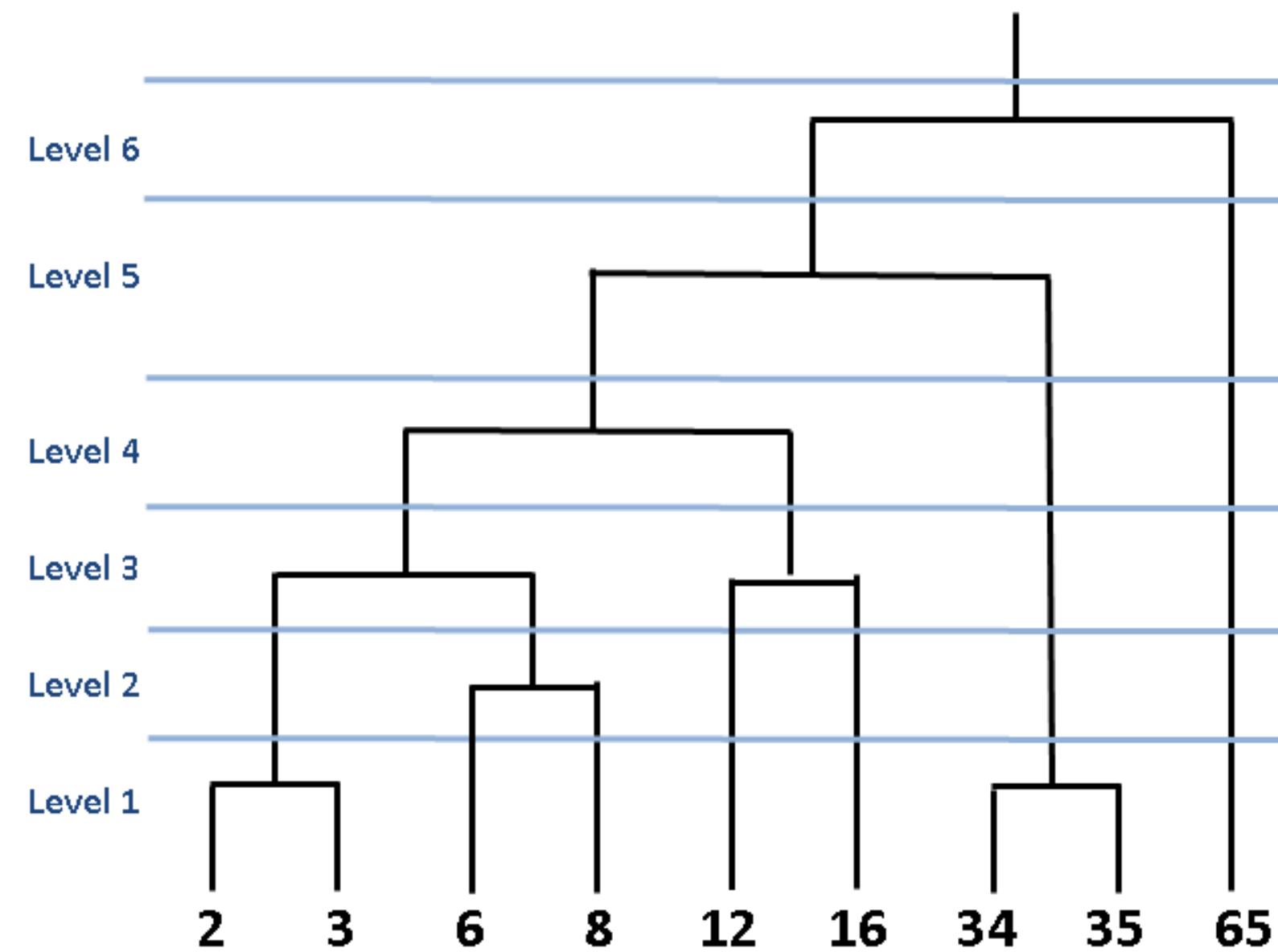
Phân cụm phân cấp (Hierarchical Clustering) là một thuật toán phân cụm không giám sát được sử dụng để nhóm các đối tượng thành các cụm dựa trên mức độ tương đồng, hoạt động bằng cách liên tục hợp nhất hoặc chia tách các cụm dựa trên một tiêu chí nhất định, tạo ra một cấu trúc phân cấp thể hiện các mối quan hệ giữa các cụm.

HIERARCHICAL

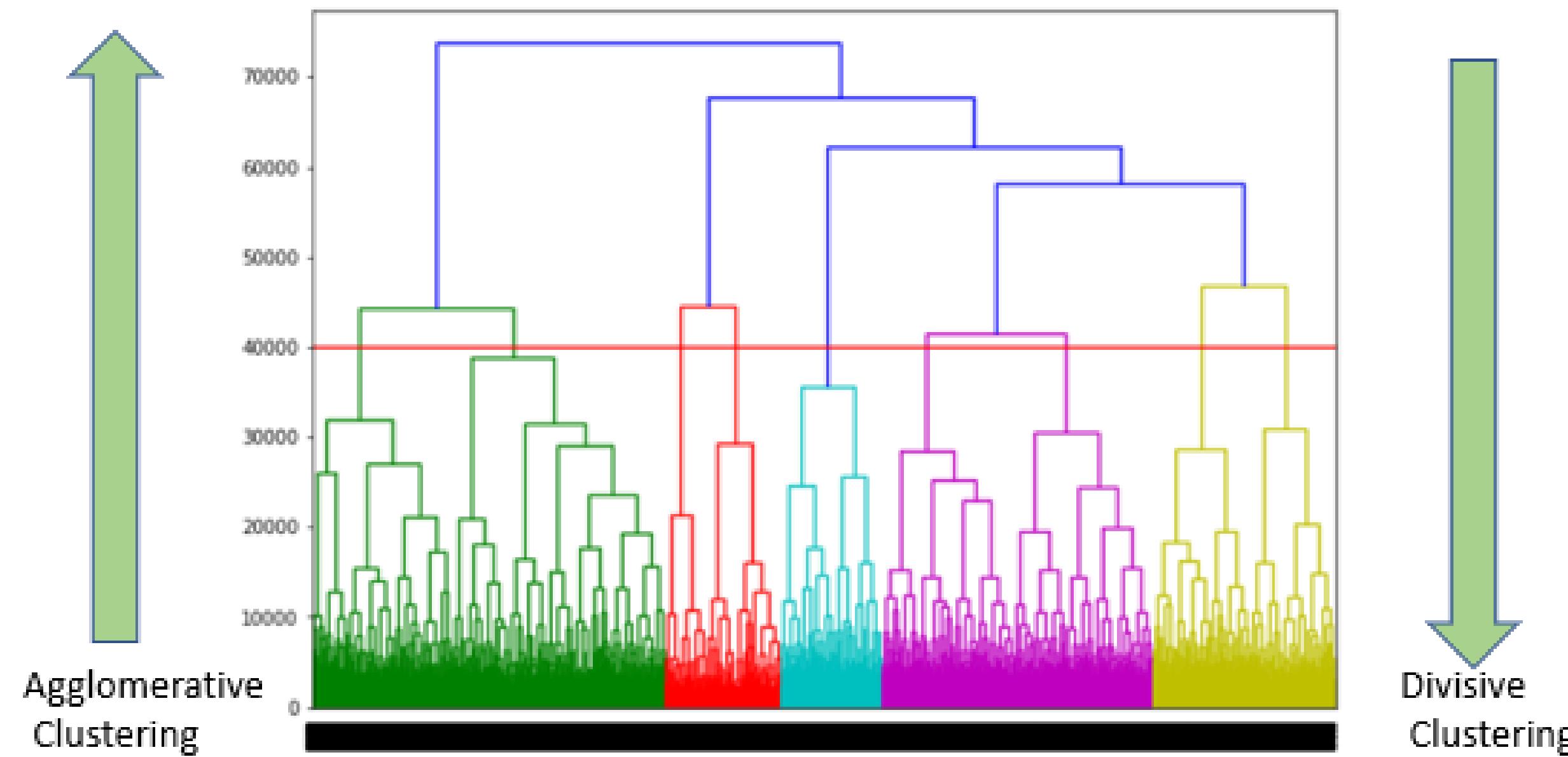
Có 2 dạng:

Agglomerative

Divisive

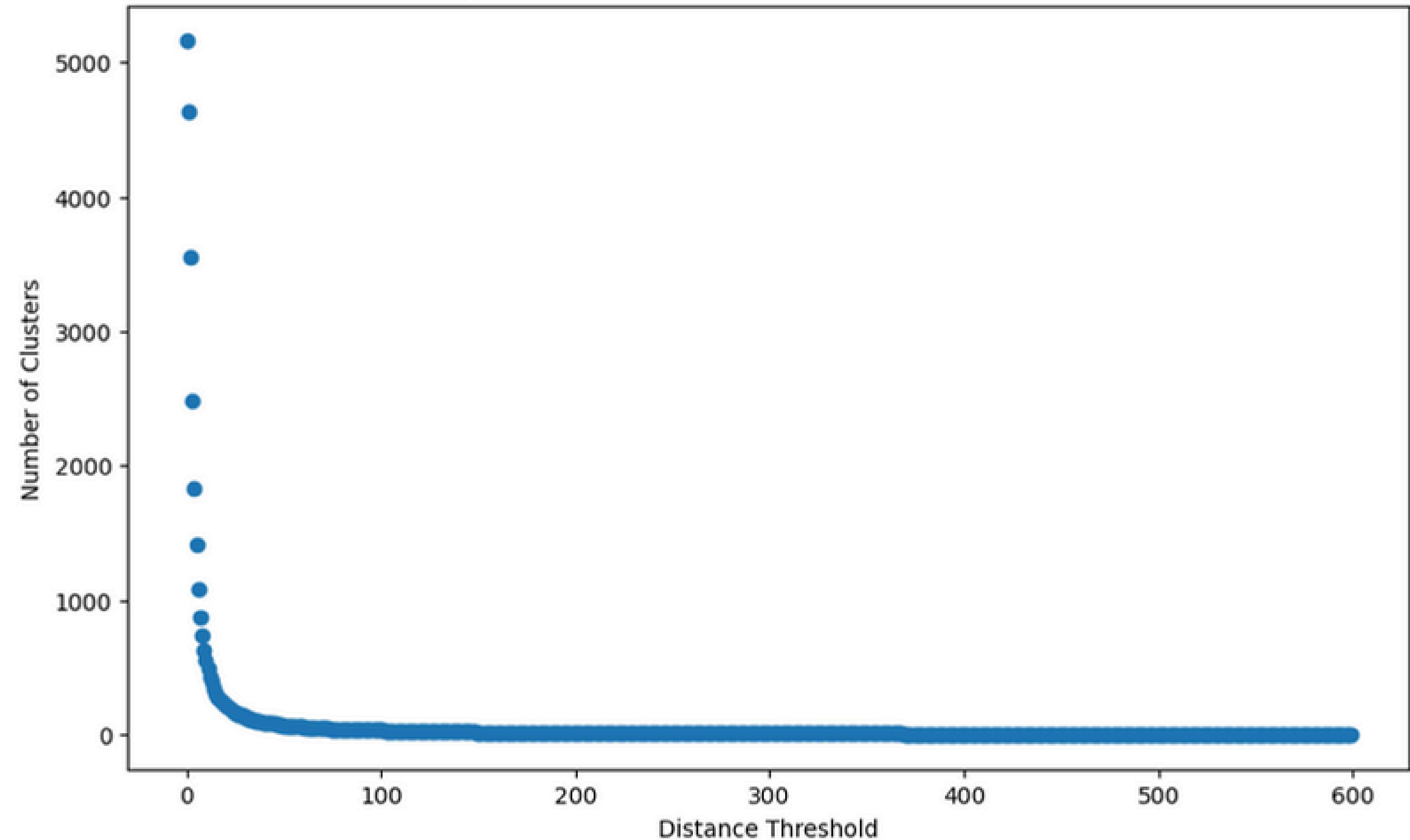


HIERARCHICAL



ỨNG DỤNG THUẬT TOÁN GOM CỤM HIERACHICAL VÀO BỘ DỮ LIỆU

Relationship between Distance Threshold and Number of Clusters



ỨNG DỤNG

THUẬT TOÁN

GOM CỤM

HIERACHICAL

VÀO

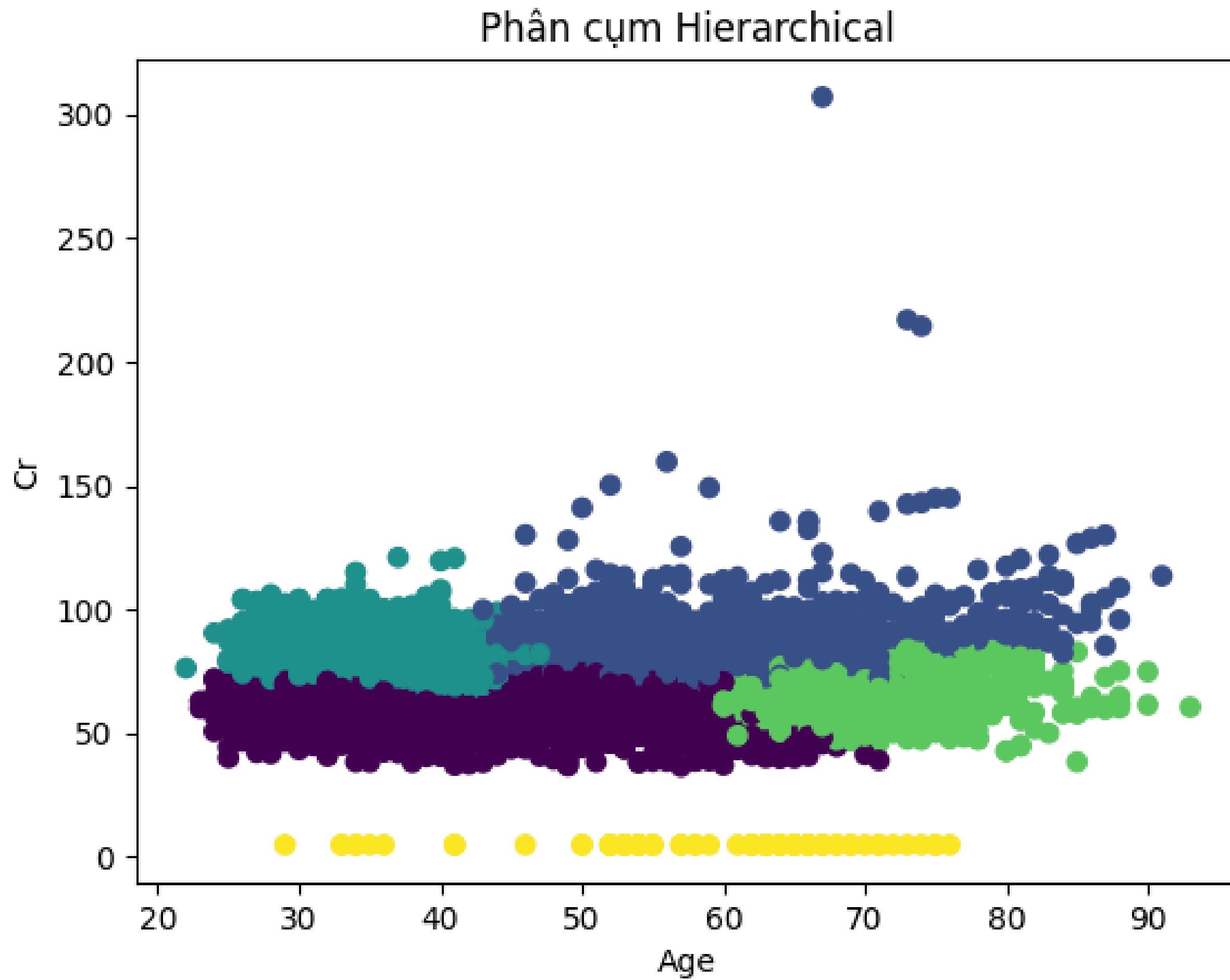
BỘ DỮ LIỆU

```
num_clusters = 5

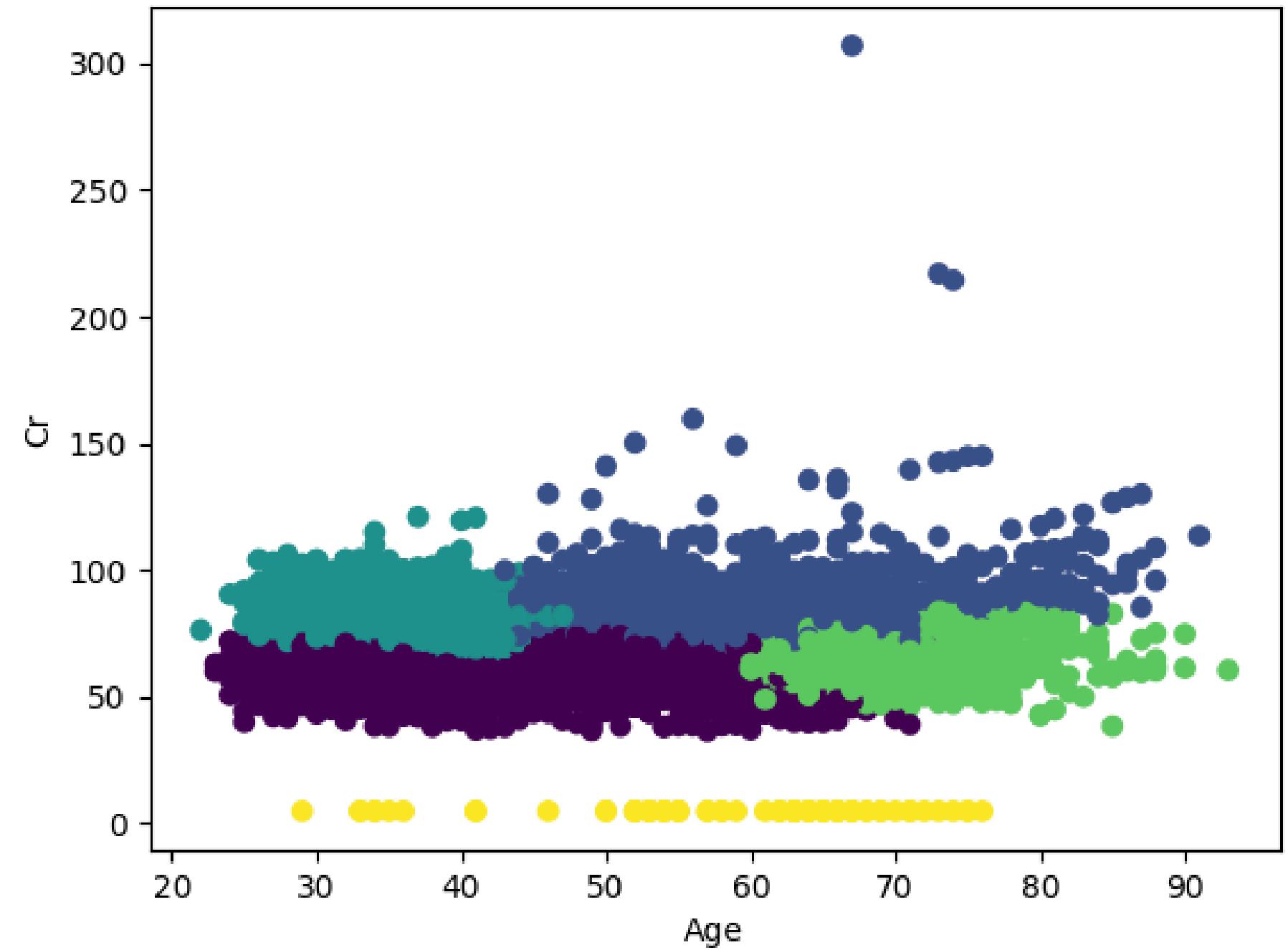
hier = AgglomerativeClustering(n_clusters = num_clusters,
                                compute_distances = True)
hier.fit(df_hier)

df_hier['Cluster'] = hier.labels_
```

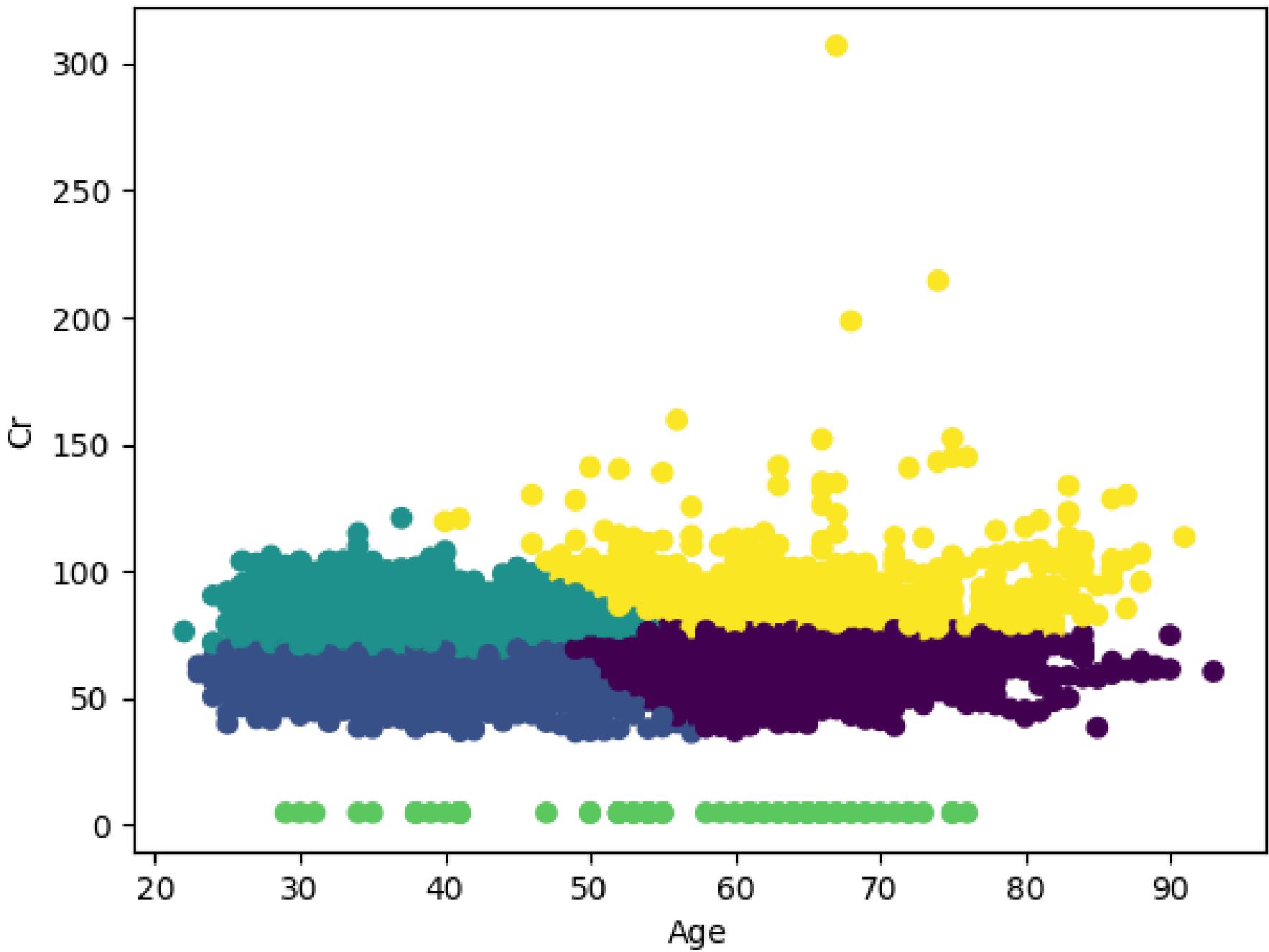
KẾT QUẢ
THU ĐƯỢC
SAU KHI
GOM CỤM
BẰNG
HIERACHICAL



Phân cụm Hierarchical



Phân cụm K-Means



PHÂN LỚP

- KNN
- Decision tree



oooo

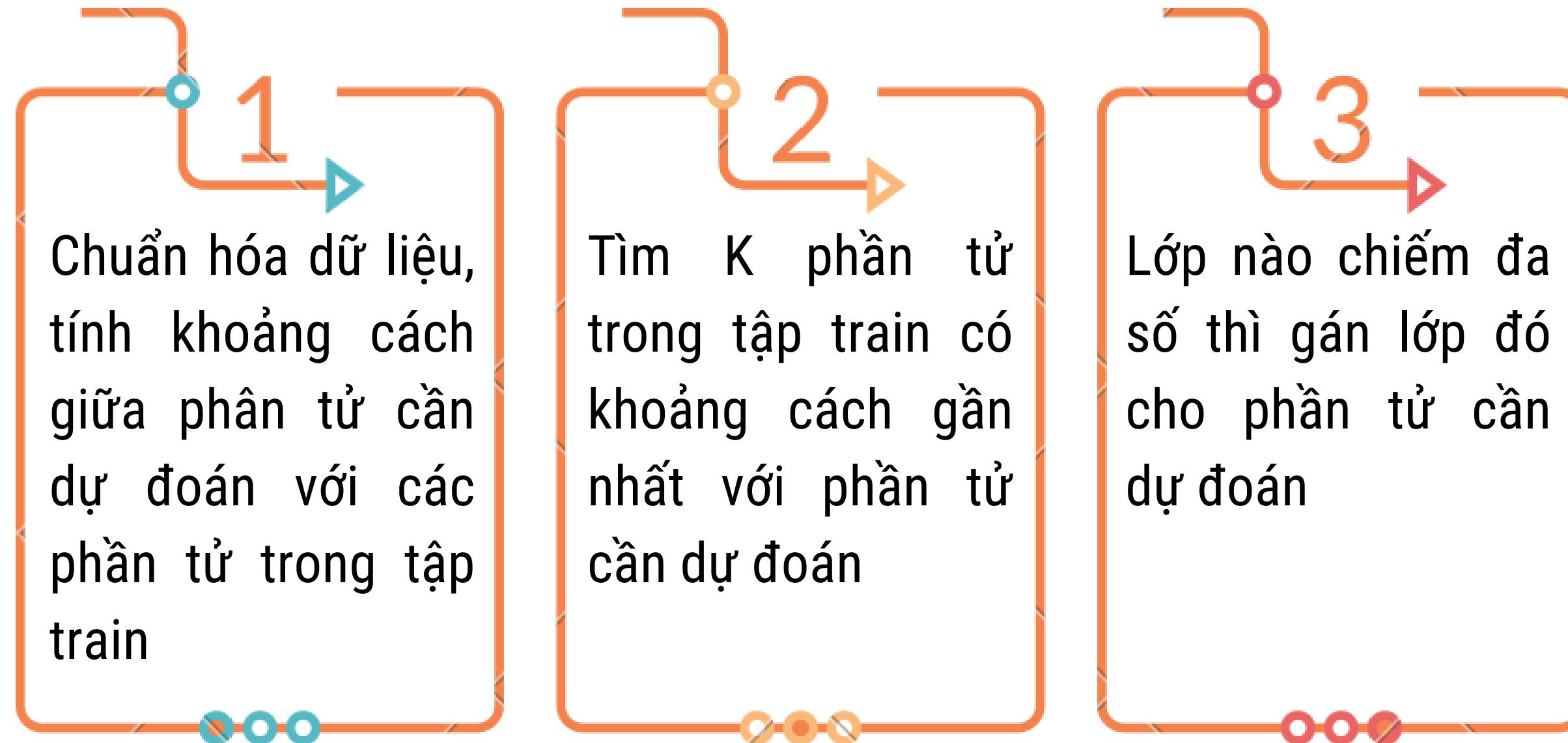
+++++
+++++
++
++
++

K-Nearest Neighbor

K-nearest neighbor là thuật toán supervised-learning đơn giản nhưng hiệu quả. Thuật toán này phân lớp cho một điểm dữ liệu mới dựa vào K điểm dữ liệu trong tập dữ liệu train có khoảng cách gần nhất so với điểm dữ liệu mới.



TÓM TẮT CÁC BƯỚC THỰC HIỆN



Age	Gender	BMI	Chol	TG	HDL	LDL	Cr	BUN	Diagnosis
26	M	23	3.7	1.4	1.1	2.1	62	4.5	0
33	M	21	4.9	1	0.8	2	46	7.1	0
58	M	33	6.6	2.9	1.1	4.3	800	20.8	1
60	F	26	4.4	2.1	1.1	2.5	72	6	1
56	M	26	4.7	1.3	0.9	3.3	60	3.5	?

○ ○ ○ ○



Ưu điểm

Các bước thực hiện đơn giản, thuật toán dễ hiểu và có độ phức tạp thấp nhưng vẫn có hiệu quả nhất định

Nhược điểm

Việc tính khoảng cách mất nhiều thời gian nếu dữ liệu lớn. Giá trị K nhỏ có thể ảnh hưởng đến độ chính xác. Cần xác định giá trị K phù hợp

ÁP DỤNG

KNN VÀO

DỰ ĐOÁN

BỆNH

TIỂU

ĐƯỜNG

Load data

[61] 1 df_knn = df_train.copy(deep=True)

[62] 1 df_knn

	Age	Gender	BMI	Chol	TG	HDL	LDL	Cr	BUN	Diagnosis
0	35	1	24.000000	5.070000	3.110000	1.030000	3.180000	81.200000	3.380000	0.0
1	30	1	23.000000	4.820000	1.180000	1.450000	2.810000	97.100000	4.800000	0.0
2	40	0	20.000000	4.600000	0.500000	1.250000	2.620000	79.000000	3.790000	0.0
3	35	1	26.000000	4.900000	2.530000	1.390000	2.080000	86.000000	6.100000	0.0
4	50	1	21.000000	4.410000	0.870000	1.100000	2.650000	81.300000	6.420000	0.0
...
5161	39	1	25.000000	5.158584	4.455930	1.040756	2.312917	78.500526	3.062004	1.0
5162	59	1	23.005422	3.946289	0.486072	4.860753	4.860753	83.895121	4.866767	1.0
5163	50	1	25.689137	5.481462	1.383708	1.227565	3.252585	83.000000	4.857228	1.0
5164	56	1	25.000000	6.149469	1.099916	1.010718	3.890666	77.028201	3.900978	1.0
5165	59	1	27.000000	4.994040	1.812976	4.860753	4.860753	70.769520	4.860753	1.0

5166 rows × 10 columns



ÁP DỤNG

KNN VÀO

DỰ ĐOÁN

BỆNH

TIỂU

ĐƯỜNG

k = 1

```
✓ [63] 1 import pandas as pd
        2 from sklearn.preprocessing import LabelEncoder, StandardScaler
        3 from sklearn.neighbors import KNeighborsClassifier
        4
        5 # Assuming df_knn is your DataFrame
        6 # Separate features (X) and target variable (y)
        7 X = df_knn[['Age', 'Gender', 'BMI', 'Chol', 'TG', 'HDL', 'LDL', 'Cr', 'BUN']]
        8 y = df_knn['Diagnosis']
        9
       10 # Standardize the features
       11 scaler = StandardScaler()
       12 X_scaled = scaler.fit_transform(X)
       13
       14 # Train the KNN model
       15 k = 1 # Number of neighbors
       16 knn_model = KNeighborsClassifier(n_neighbors=k)
       17 knn_model.fit(X_scaled, y)
```

▼ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1)

ÁP DỤNG KNN VÀO DỰ ĐOÁN BỆNH TIỂU ĐƯỜNG

```
[ ] 1 # Assuming df_test is your test DataFrame
2 # Separate features (X_test) and target variable (y_test)
3 X_test = df_test[['Age', 'Gender', 'BMI', 'Chol', 'TG', 'HDL', 'LDL', 'Cr', 'BUN']]
4 y_test = df_test['Diagnosis']
5
6 # Standardize the features using the same scaler as used for training
7 X_test_scaled = scaler.transform(X_test)
8
9 # Predict on the test set
10 y_pred = knn_model.predict(X_test_scaled)
11
12 # Evaluate the performance of the model
13 accuracy = knn_model.score(X_test_scaled, y_test)
14 print("Accuracy:", accuracy)
15
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:
  and should_run_async(code)
Accuracy: 0.6252631578947369
```

ÁP DỤNG

KNN VÀO

DỰ ĐOÁN

BỆNH

TIỂU

ĐƯỜNG

k = 3

```
[ ] 1 import pandas as pd
2 from sklearn.preprocessing import LabelEncoder, StandardScaler
3 from sklearn.neighbors import KNeighborsClassifier
4
5 # Assuming df_knn is your DataFrame
6 # Separate features (X) and target variable (y)
7 X = df_knn[['Age', 'Gender', 'BMI', 'Chol', 'TG', 'HDL', 'LDL', 'Cr', 'BUN']]
8 y = df_knn['Diagnosis']
9
10 # Standardize the features
11 scaler = StandardScaler()
12 X_scaled = scaler.fit_transform(X)
13
14 # Train the KNN model
15 k = 3 # Number of neighbors
16 knn_model = KNeighborsClassifier(n_neighbors=k)
17 knn_model.fit(X_scaled, y)
```

/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `

and should_run_async(code)

+

KNeighborsClassifier

KNeighborsClassifier(n_neighbors=3)

ÁP DỤNG KNN VÀO DỰ ĐOÁN BỆNH TIỂU ĐƯỜNG

```
[ ] 1 # Assuming df_test is your test DataFrame
2 # Separate features (x_test) and target variable (y_test)
3 x_test = df_test[['Age', 'Gender', 'BMI', 'Chol', 'TG', 'HDL', 'LDL', 'Cr', 'BUN']]
4 y_test = df_test['Diagnosis']
5
6 # Standardize the features using the same scaler as used for training
7 x_test_scaled = scaler.transform(x_test)
8
9 # Predict on the test set
10 y_pred = knn_model.predict(x_test_scaled)
11
12 # Evaluate the performance of the model
13 accuracy = knn_model.score(x_test_scaled, y_test)
14 print("Accuracy:", accuracy)
15
```

```
Accuracy: 0.6842105263157895
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `sh
and should_run_async(code)
```

ÁP DỤNG

KNN VÀO

DỰ ĐOÁN

BỆNH

TIỂU

ĐƯỜNG

k = 5

```
[ ] 1 import pandas as pd
2 from sklearn.preprocessing import LabelEncoder, StandardScaler
3 from sklearn.neighbors import KNeighborsClassifier
4
5 # Assuming df_knn is your DataFrame
6 # Separate features (X) and target variable (y)
7 X = df_knn[['Age', 'Gender', 'BMI', 'Chol', 'TG', 'HDL', 'LDL', 'Cr', 'BUN']]
8 y = df_knn['Diagnosis']
9
10 # Standardize the features
11 scaler = StandardScaler()
12 X_scaled = scaler.fit_transform(X)
13
14 # Train the KNN model
15 k = 5 # Number of neighbors
16 knn_model = KNeighborsClassifier(n_neighbors=k)
17 knn_model.fit(X_scaled, y)
```

/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should

and should_run_async(code)

▪ KNeighborsClassifier

KNeighborsClassifier()

ÁP DỤNG

KNN VÀO

DỰ ĐOÁN

BỆNH

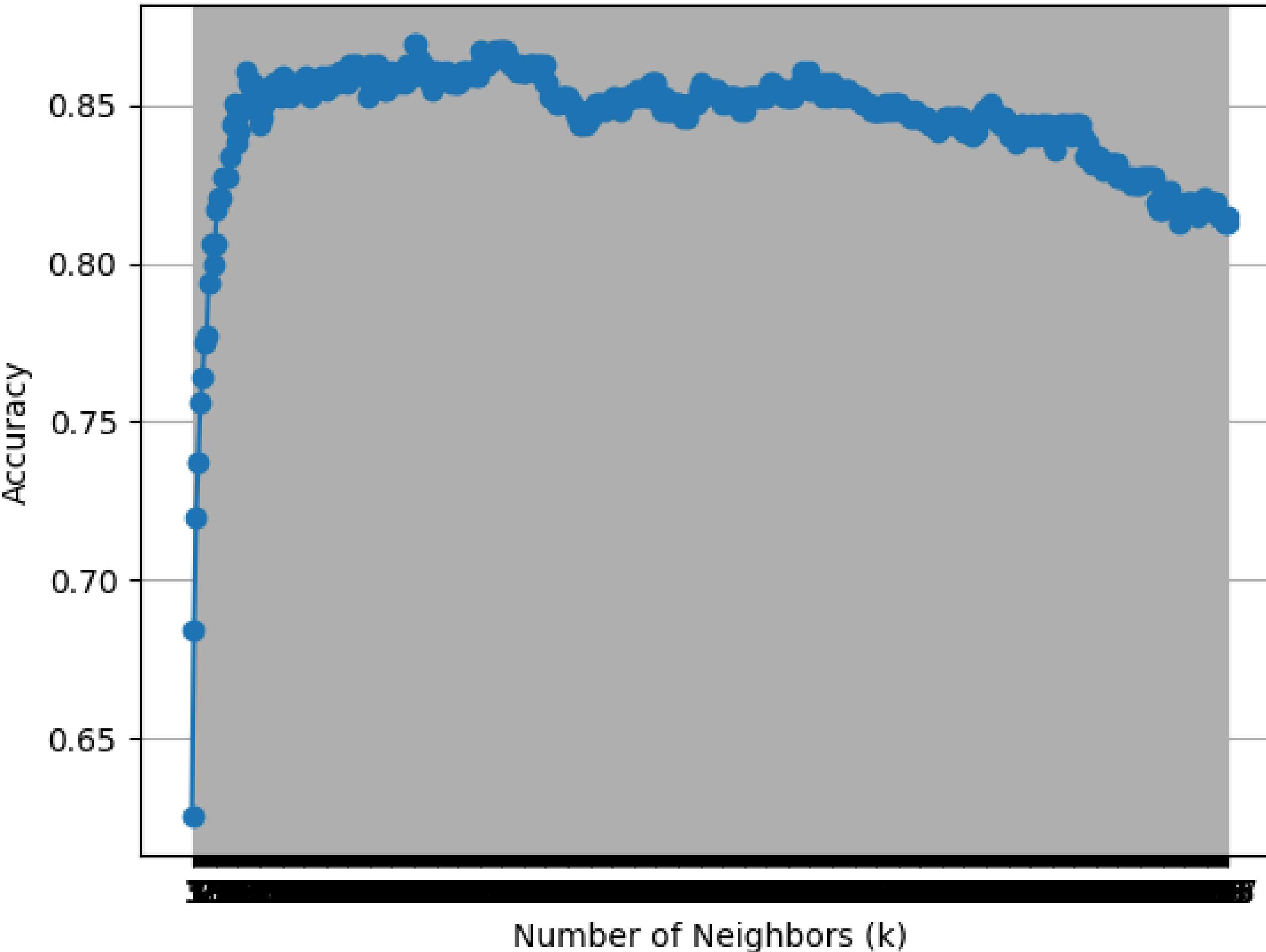
TIỂU

ĐƯỜNG

```
1 from sklearn.preprocessing import LabelEncoder, StandardScaler
2 from sklearn.neighbors import KNeighborsClassifier
3
4 # Initialize empty list to store accuracies
5 accuracies = []
6
7 # Range of k values to test
8 k_values = range(1, 999) # Test k from 1 to 999
9
10 # Assuming df_knn is your DataFrame
11 # Separate features (X) and target variable (y)
12 X = df_knn[['Age', 'Gender', 'BMI', 'Chol', 'TG', 'HDL', 'LDL', 'Cr', 'BUN']]
13 y = df_knn['Diagnosis']
14
15 # Standardize the features
16 scaler = StandardScaler()
17 X_scaled = scaler.fit_transform(X)
18
19 # Assuming df_test is your test DataFrame
20 # Separate features (X_test) and target variable (y_test)
21 X_test = df_test[['Age', 'Gender', 'BMI', 'Chol', 'TG', 'HDL', 'LDL', 'Cr', 'BUN']]
22 y_test = df_test['Diagnosis']
23
24 # Standardize the features using the same scaler as used for training
25 X_test_scaled = scaler.transform(X_test)
26
27 # Iterate over each value of k
28 for k in k_values:
29     knn_model = KNeighborsClassifier(n_neighbors=k)
30     knn_model.fit(X_scaled, y)
31     accuracies.append(knn_model.score(X_test_scaled, y_test)) # Evaluate model accuracy on training set
32
33
34 # Plot accuracy as a function of k
35 plt.plot(k_values, accuracies, marker='o')
36 plt.title('Accuracy vs. Number of Neighbors (k)')
37 plt.xlabel('Number of Neighbors (k)')
38 plt.ylabel('Accuracy')
39 plt.xticks(k_values)
40 plt.grid(True)
41 plt.show()
```

**ÁP DỤNG
KNN VÀO
DỰ ĐOÁN
BỆNH
TIỂU
ĐƯỜNG**

Accuracy vs. Number of Neighbors (k)



ÁP DỤNG

KNN VÀO

DỰ ĐOÁN

BỆNH

TIỂU

ĐƯỜNG

```
▶ 1 index_of_max_accuracy = np.argmax(accuracies)
  2 print("Maximum accuracy:", max(accuracies))
  3 print("Index of maximum accuracy:", index_of_max_accuracy)
  4

Maximum accuracy: 0.8694736842105263
Index of maximum accuracy: 107
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not
and should_run_async(code)
```

k = 215

```
▶ 1 from sklearn.preprocessing import LabelEncoder, StandardScaler
  2 from sklearn.neighbors import KNeighborsClassifier
  3
  4 # Assuming df_knn is your DataFrame
  5 # Separate features (X) and target variable (y)
  6 X = df_knn[['Age', 'Gender', 'BMI', 'Chol', 'TG', 'HDL', 'LDL', 'Cr', 'BUN']]
  7 y = df_knn['Diagnosis']
  8
  9 # Standardize the features
10 scaler = StandardScaler()
11 X_scaled = scaler.fit_transform(X)
12
13 # Train the KNN model
14 k = 215 # Number of neighbors
15 knn_model = KNeighborsClassifier(n_neighbors=k)
16 knn_model.fit(X_scaled, y)
```

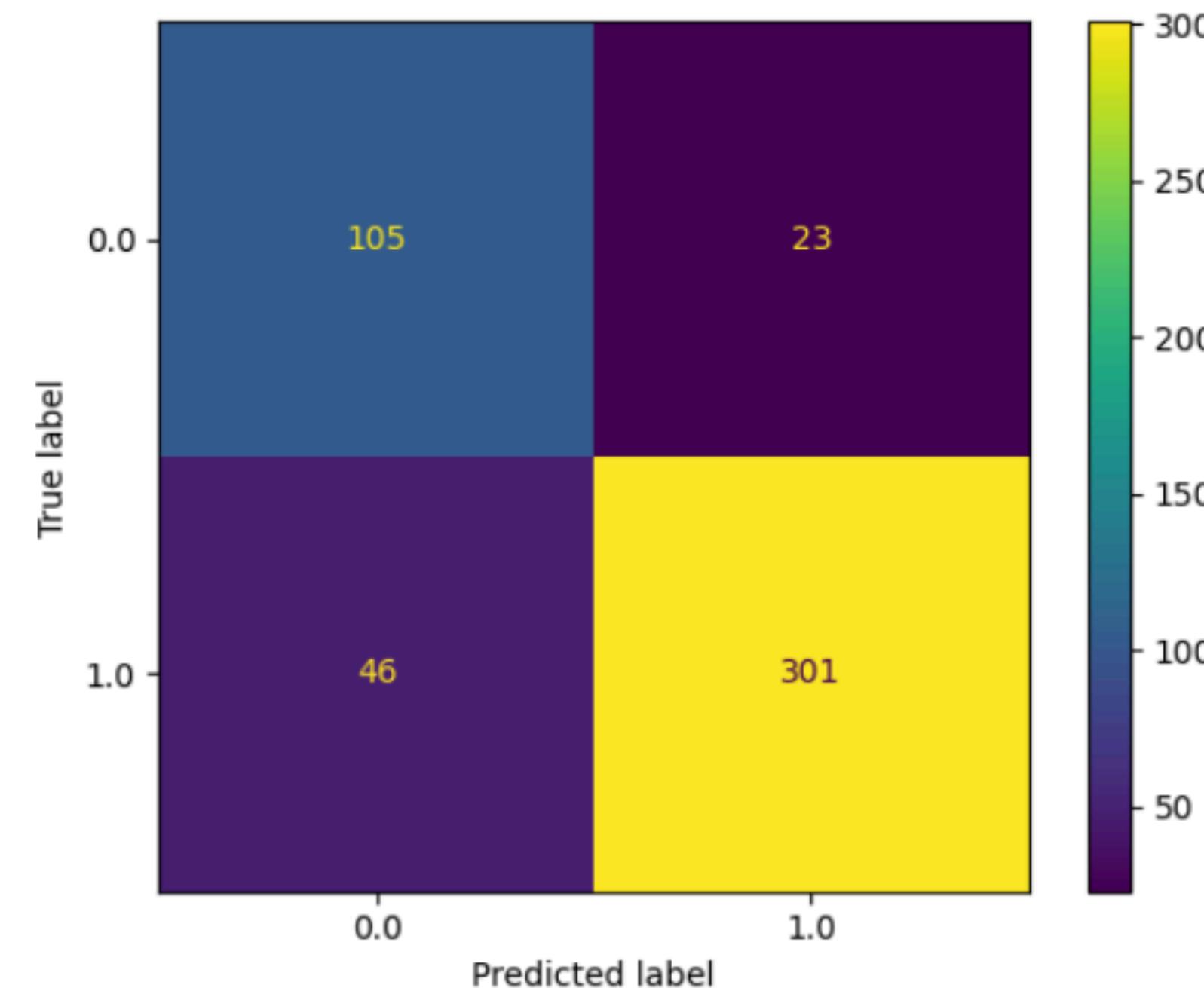
ÁP DỤNG KNN VÀO DỰ ĐOÁN BỆNH TIỂU ĐƯỜNG

```
[ ] 1 # Assuming df_test is your test DataFrame
2 # Separate features (X_test) and target variable (y_test)
3 X_test = df_test[['Age', 'Gender', 'BMI', 'Chol', 'TG', 'HDL', 'LDL', 'Cr', 'BUN']]
4 y_test = df_test['Diagnosis']
5
6 # Standardize the features using the same scaler as used for training
7 X_test_scaled = scaler.transform(X_test)
8
9 # Predict on the test set
10 y_pred = knn_model.predict(X_test_scaled)
11
12 # Evaluate the performance of the model
13 accuracy = knn_model.score(X_test_scaled, y_test)
14 print("Accuracy:", accuracy)
15
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_r
    and should_run_async(code)
Accuracy: 0.8694736842105263
```

ÁP DỤNG KNN VÀO DỰ ĐOÁN BỆNH TIỂU ĐƯỜNG

```
1 from sklearn.metrics import ConfusionMatrixDisplay  
2 ConfusionMatrixDisplay.from_predictions(label_test, y_pred)  
3 plt.show()
```



```
1 from sklearn.metrics import confusion_matrix  
2 from tabulate import tabulate  
3  
4 TN, FP, FN, TP = confusion_matrix(label_test, y_pred).ravel()  
  
+-----+  
| Metric | Value |  
+=====+  
| Recall | 0.887608 |  
| Precision | 0.930514 |  
| False Positive Rate (FPR) | 0.179688 |  
| True Negative Rate (TNR) | 0.820312 |  
| False Negative Rate (FNR) | 0.112392 |  
+-----+
```

o o o o

+ + + + +
+ + + + +
+ +
+ +
+ +

Decision Tree

Cây quyết định là một mô hình có cấu trúc như flowchart. Mỗi node thể hiện một điều kiện. Các kết quả của điều kiện đó được chia thành các nhánh. Các node không có con gọi là node lá, biểu diễn kết quả phân loại



oooo

+++++

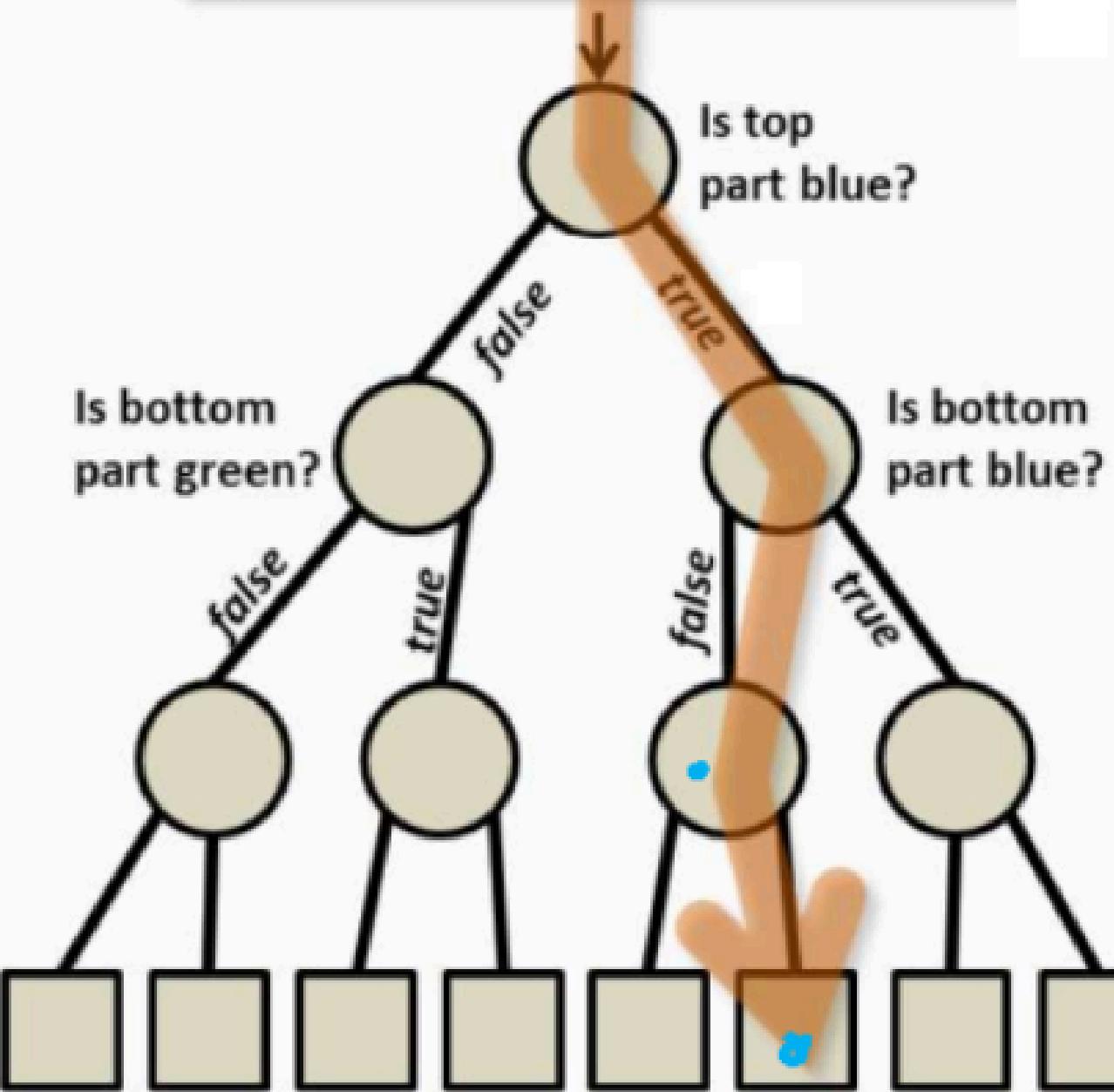
+++++

+++++

+++++

+++++

A decision tree



○ ○ ○ ○



Ưu điểm

Mô hình đơn giản và dễ hiểu, không cần nhiều dữ liệu train. Tốc độ xây dựng mô hình và phân lớp nhanh. Áp dụng được cho nhiều dạng dữ liệu khác nhau.

Nhược điểm

Mô hình phụ thuộc vào dữ liệu train, nếu dữ liệu thay đổi phải xây dựng lại mô hình mới. Kết quả của thuật toán không đảm bảo chính xác hoàn toàn.

ÁP DỤNG

DECISION

TREE VÀO

DỰ ĐOÁN

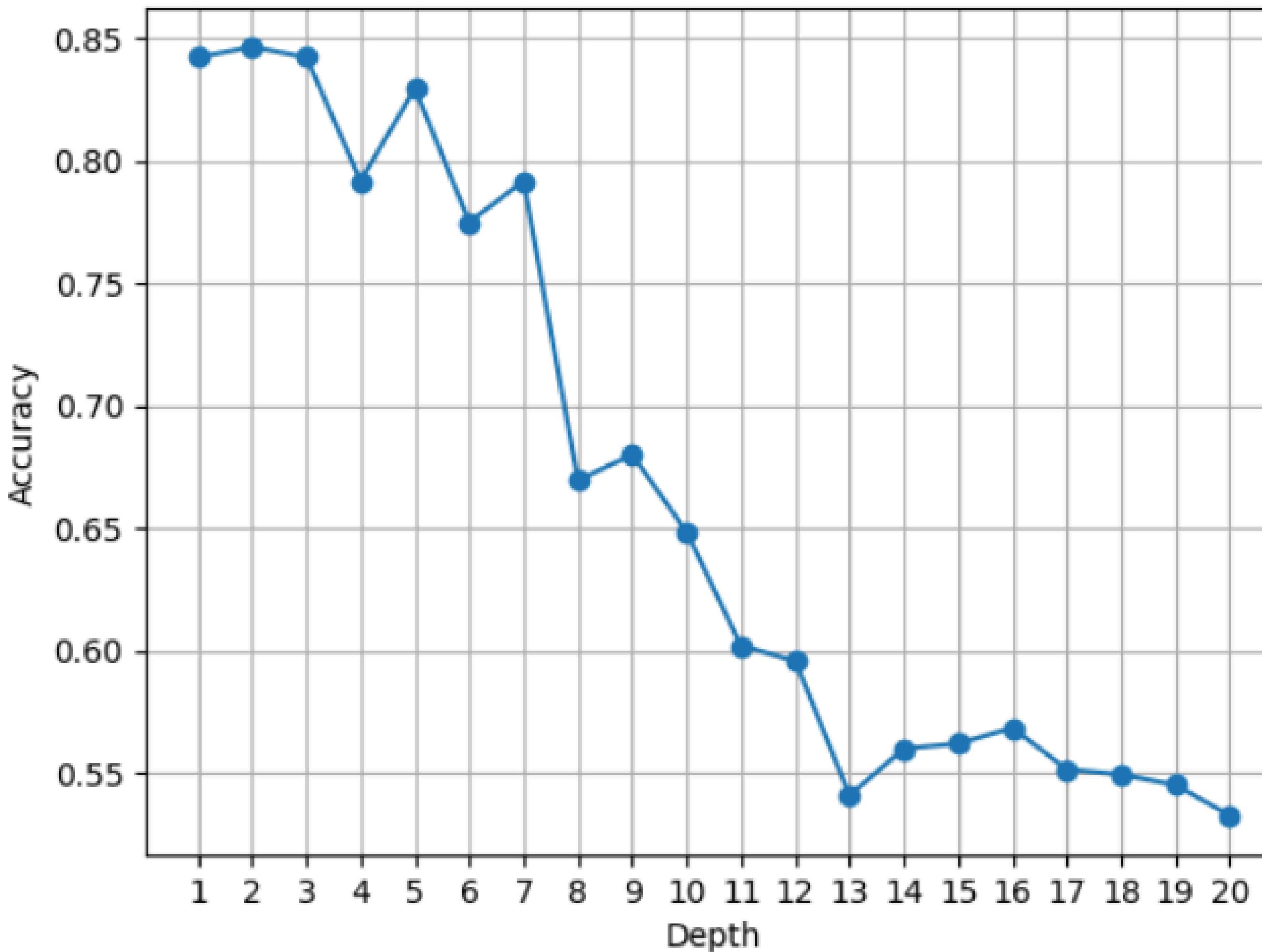
BỆNH

TIỂU

ĐƯỜNG

```
1 from sklearn import tree
2 from sklearn.tree import DecisionTreeClassifier
3 from sklearn.metrics import accuracy_score
4
5 # Assuming df_knn is your DataFrame
6 # Splitting the data into features (X) and target variable (y)
7 d = df_knn.drop('Diagnosis', axis=1)
8 lb = df_knn['Diagnosis']
9
10 # Assuming df_test is your test DataFrame
11 # Separate features (X_test) and target variable (y_test)
12 d_test = df_test[['Age', 'Gender', 'BMI', 'Chol', 'TG', 'HDL', 'LDL', 'Cr', 'BUN']]
13 lb_test = df_test['Diagnosis']
14
15 tree_accuracies = []
16
17 depth_values = range(1, 21)
18 for depth in depth_values:
19     clf = DecisionTreeClassifier(max_depth=depth)
20     clf.fit(d, lb)
21     y_pred_test = clf.predict(d_test)
22     test_accuracy = accuracy_score(lb_test, y_pred_test)
23     tree_accuracies.append(test_accuracy)
24
25 # Plot accuracy as a function of k
26 plt.plot(depth_values, tree_accuracies, marker='o')
27 plt.title('Accuracy vs Depth')
28 plt.xlabel('Depth')
29 plt.ylabel('Accuracy')
30 plt.xticks(depth_values)
31 plt.grid(True)
32 plt.show()
```

Accuracy vs Depth



ÁP DỤNG
DECISION
TREE VÀO
DỰ ĐOÁN
BỆNH
TIỂU
ĐƯỜNG

ÁP DỤNG

DECISION

TREE VÀO

DỰ ĐOÁN

BỆNH

TIỂU

ĐƯỜNG

```
→ /usr/local/lib/python3.10/dist-packages/ipykernel/ipkern  
and should_run_async(code)
```

```
▼ DecisionTreeClassifier
```

```
DecisionTreeClassifier(max_depth=2)
```

```
[ ] 1 # Making predictions  
2 y_pred_test = clf.predict(d_test)  
3  
4 # Evaluating the model  
5 test_accuracy = accuracy_score(lb_test, y_pred_test)  
6  
7 print("Testing accuracy:", test_accuracy)
```

```
→ Testing accuracy: 0.8463157894736842
```

ÁP DỤNG

DECISION

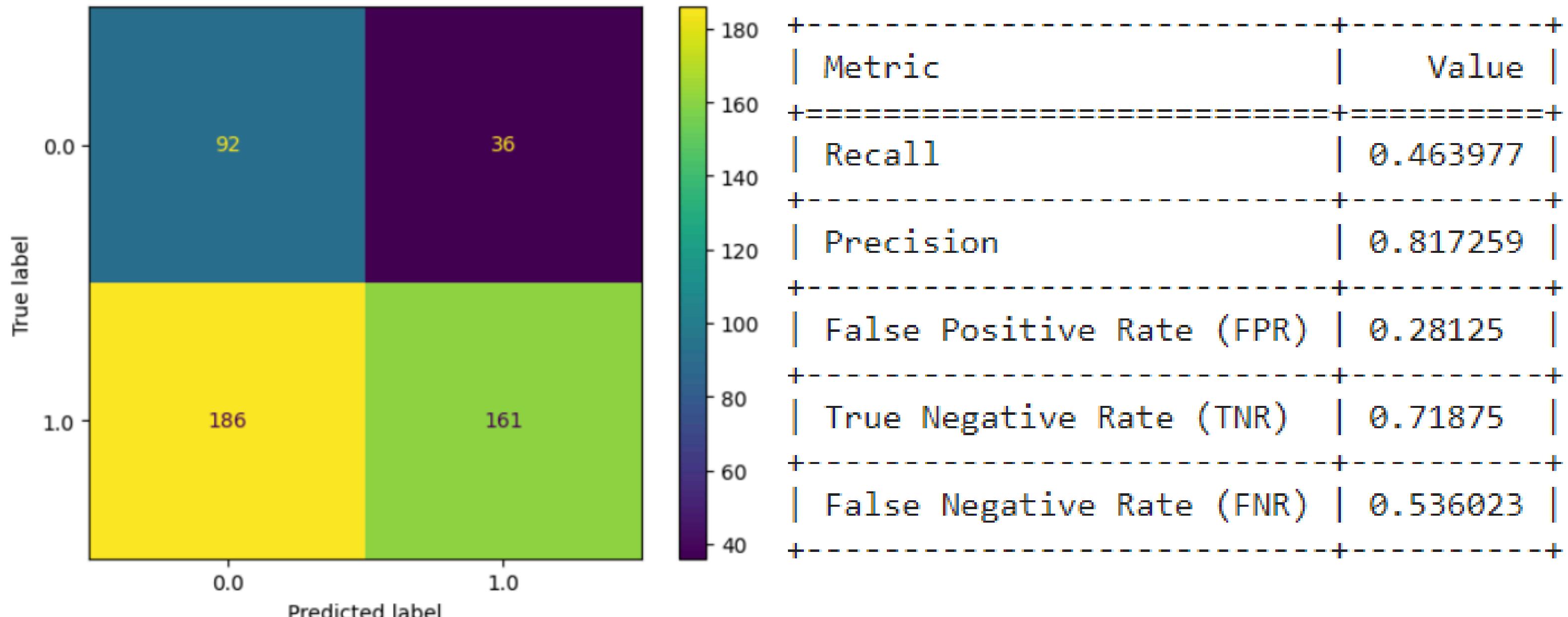
TREE VÀO

DỰ ĐOÁN

BỆNH

TIỂU

ĐƯỜNG



KẾT LUẬN

- Chẩn đoán được hầu hết người có bệnh tiểu đường (Precision)
- Xác định được phần lớn người không mắc bệnh (TNR)
- FPR và FNR không đáng kể

K-Nearest Neighbor

- Chẩn đoán được phần lớn người có bệnh tiểu đường (Precision)
- Xác định được phần lớn người không mắc bệnh (TNR)
- FPR và FNR không đáng kể

Metric	Value
Recall	0.887608
Precision	0.930514
False Positive Rate (FPR)	0.179688
True Negative Rate (TNR)	0.820312
False Negative Rate (FNR)	0.112392

Metric	Value
Recall	0.463977
Precision	0.817259
False Positive Rate (FPR)	0.28125
True Negative Rate (TNR)	0.71875
False Negative Rate (FNR)	0.536023

Decision Tree

- Chẩn đoán được phần lớn người có bệnh tiểu đường (Precision)
- Xác định được đa số người không mắc bệnh nhưng chưa đáng kể (TNR)
- Chẩn đoán nhiều người bệnh thật là không mắc bệnh, gây hậu quả nghiêm trọng (FNR)

○○○○

Thank You!

