



CHRONIC KIDNEY DISEASE PREDICTION

PRESENTED BY : ALEX TRAN, PETER LE





Mục Lục



1. Tổng quan về đề tài
2. EDA
3. Datapreprocessing
4. Train model
5. Đánh giá sơ bộ
6. Chứng minh kết quả
7. Kết luận





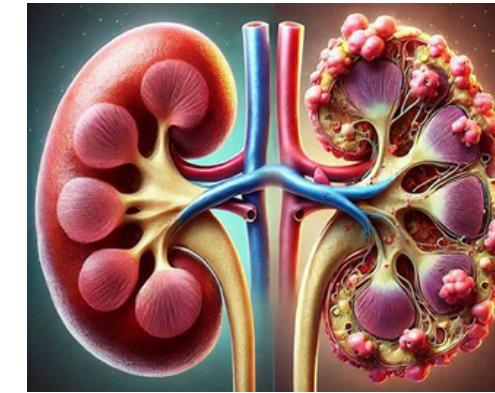
Tổng quan về đề tài



Lý do chọn đề tài

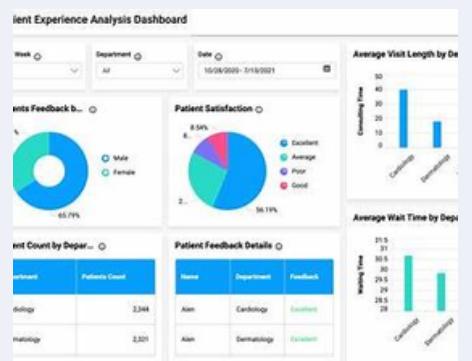
1. CKD là bệnh nguy hiểm & âm thầm

CKD tiến triển chậm, khó nhận biết ở giai đoạn đầu và dễ dẫn đến suy thận nếu phát hiện muộn.



2. Chẩn đoán truyền thống còn hạn chế

Dựa trên nhiều chỉ số rời rạc, tính chủ quan cao và khó sàng lọc hiệu quả trên số lượng lớn bệnh nhân.



3. Cần dự đoán sớm dựa trên dữ liệu

Dữ liệu lâm sàng cho phép phát hiện sớm nguy cơ CKD, hỗ trợ theo dõi và quản lý bệnh nhân tốt hơn.

4. Machine Learning là giải pháp phù hợp

ML giúp phân tích nhiều biến cùng lúc và đưa ra dự đoán nhanh, chính xác, khách quan hơn.



5. Ý nghĩa khoa học & thực tiễn

Nghiên cứu thúc đẩy ứng dụng AI trong y tế và hỗ trợ bác sĩ cải thiện hiệu quả chẩn đoán CKD.

Bệnh thận mạn tính là gì?

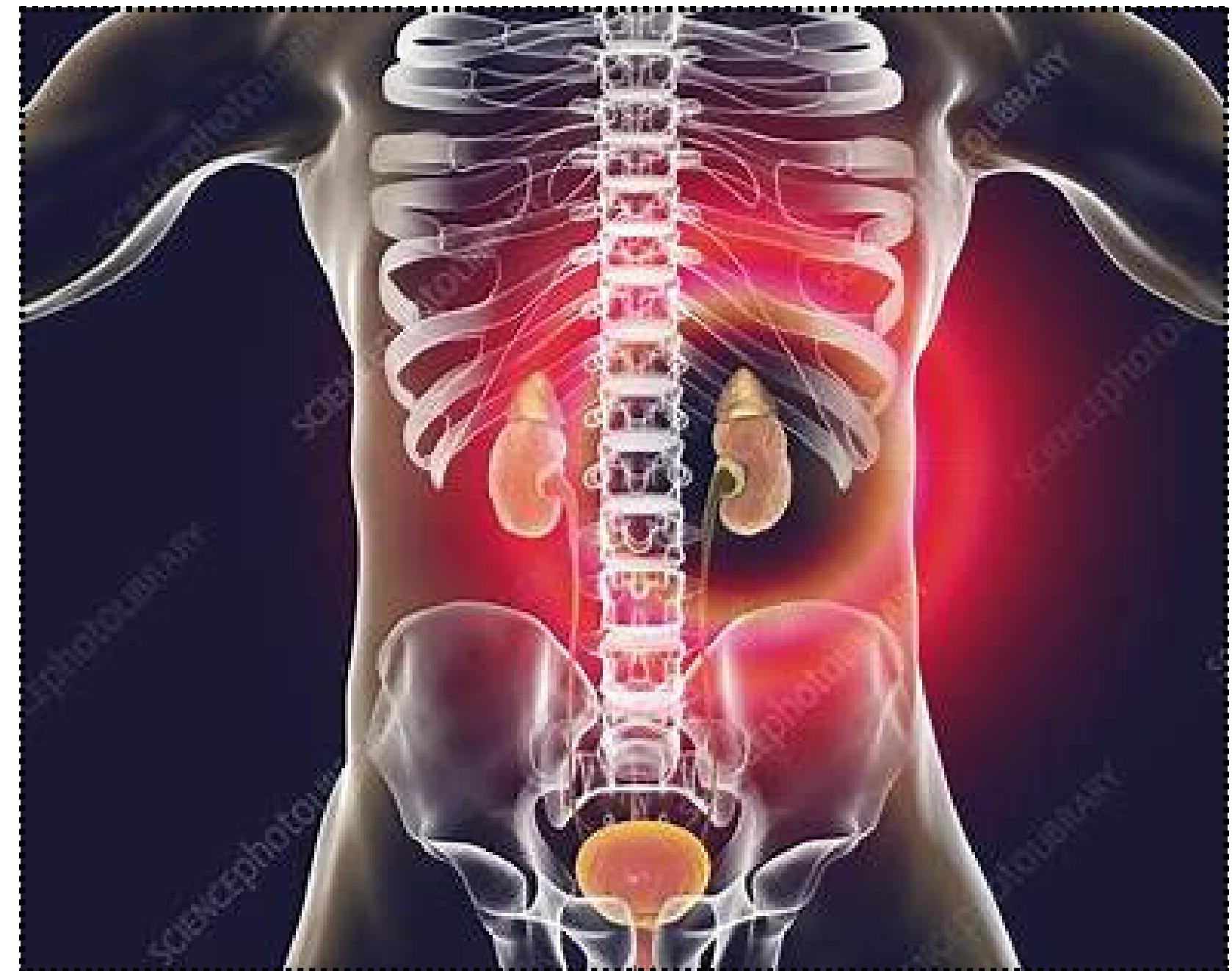
CKD (Chronic Kidney Disease) = Suy giảm chức năng thận kéo dài ≥ 3 tháng, biểu hiện bởi:

- Giảm mức lọc cầu thận (GFR < 60 ml/min/1.73m²)
- Hoặc có dấu hiệu tổn thương thận: protein niệu, tiểu máu, bất thường hình ảnh học

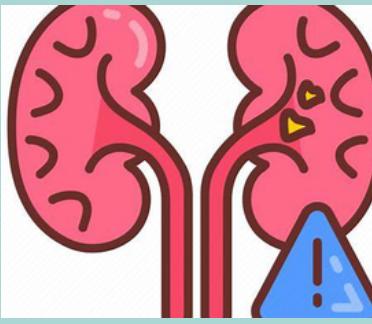
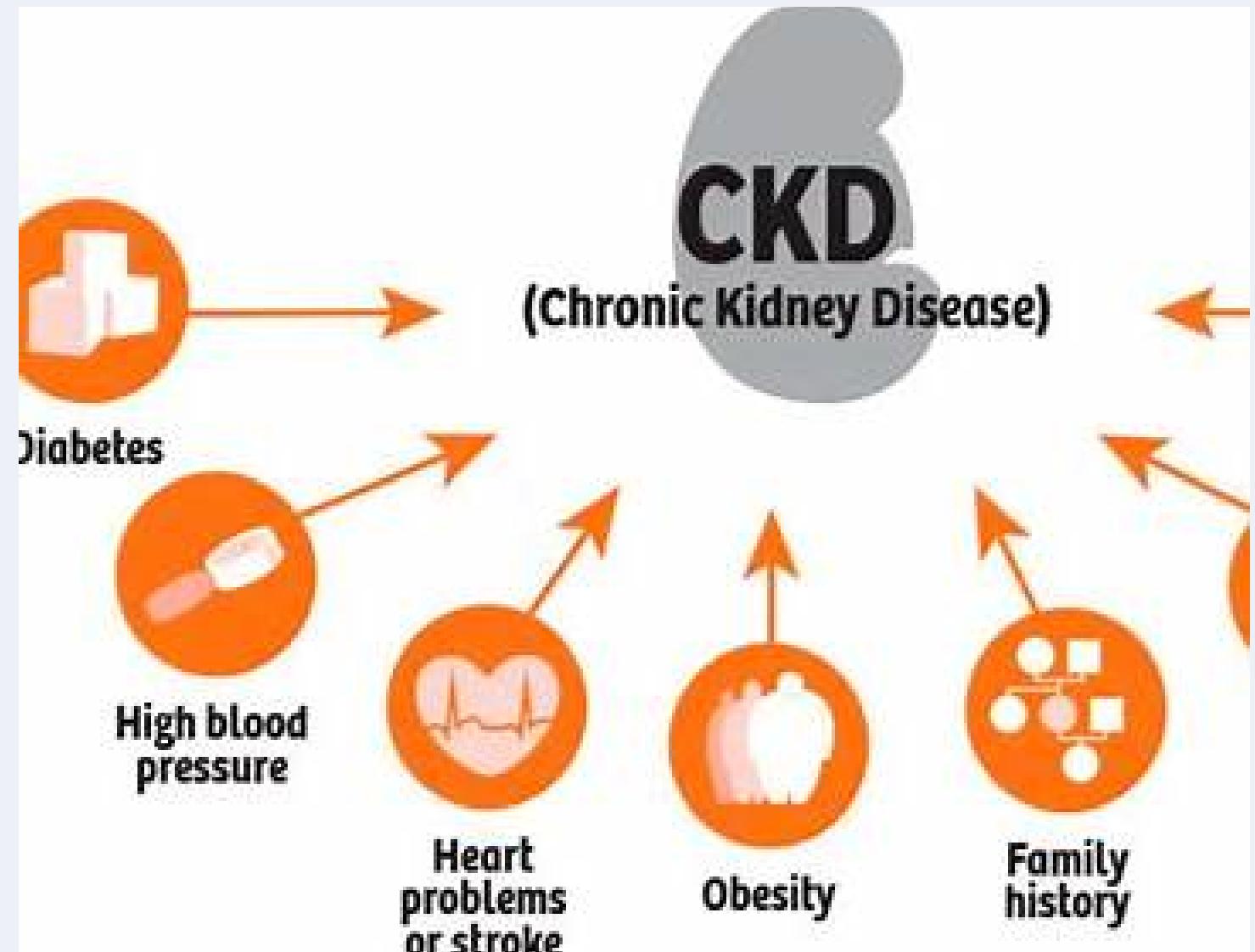
CKD là bệnh mạn tính, tiến triển rất âm thầm và gần như không thể hồi phục khi ở giai đoạn muộn của bệnh

CKD gây nhiều biến chứng nguy hiểm:

- Biến chứng tim mạch như suy tim
- Rối loạn huyết học
- Rối loạn điện giải & chuyển hóa
- Suy thận giai đoạn cuối (ESRD)
- Gây nguy hiểm trực tiếp đến tính mạng



Tại sao cần dự đoán CKD sớm?



CKD tiến triển âm thầm ở những giai đoạn đầu



Điều trị sớm làm chậm tiến triển và có thể bảo tồn thận



Ngăn chặn kịp thời biến chứng tim mạch, thiếu máu , ESRD



Giảm chi phí điều trị (lọc máu, ghép thận) , và giảm thiểu tài nguyên cơ sở y tế

Mục tiêu nghiên cứu

1. Mục tiêu chính

Xây dựng mô hình dự đoán sớm và xác nhận nguy cơ mắc bệnh thận mạn tính (CKD) dựa trên dữ liệu bệnh nhân. Với độ nhạy và chính xác cao

2. Phạm vi và phương pháp

Ứng dụng 4 thuật toán Machine Learning phổ biến:

- Logistic Regression
- Decision Tree
- Random Forest
- K-Nearest Neighbors (KNN)

3. Mục tiêu so sánh

So sánh hiệu suất giữa các mô hình để xác định thuật toán phù hợp nhất cho bài toán dự đoán CKD.

4. Tiêu chí đánh giá

Đánh giá mô hình theo các chỉ số:

- Accuracy
- Confusion matrix
- Roc&AUC
- K-fold cv

5. Kỳ vọng

Thu được mô hình tối ưu có độ nhạy, độ chính xác cao và ổn định để hỗ trợ sàng lọc sớm trong y tế.

Tổng quan về dataset

Tên: Chronic Kidney Disease Dataset

Số mẫu: 400 bệnh nhân

Nguồn gốc: Được thu thập trong 2 tháng ở bệnh viện Fortis Gurgaon tại Ấn Độ

Số biến: 25 biến (bao gồm numeric + categorical)

Mục tiêu: Dự đoán bệnh CKD (Mắc) hoặc không CKD (Không mắc) thận mãn tính

Dạng dữ liệu: chứa cả số (numeric) và phân loại (categorical) và không có mix data

Các cột category và numeric được phân bố khá đồng đều

Tổng quan về dataset

Danh sách các Feature : Columns

age	Tuổi	sc	Creatinine huyết thanh
bp	Huyết áp	sod	Natri
sg	Tỷ trọng nước tiểu	pot	Kali
al	Hàm lượng albumin trong nước tiểu	hemo	Huyết sắc tố
su	Mức đường trong nước tiểu	pcv	Thể tích hồng cầu đóng gói
rbc	Tình trạng hồng cầu	wc	Số lượng bạch cầu
pc	Tình trạng bạch cầu mủ	rc	Số lượng hồng cầu
pcc	Cụm bạch cầu mủ	htn	Tăng huyết áp
ba	Vi khuẩn trong nước tiểu	dm	Tiểu đường
bgr	Đường huyết ngẫu nhiên	cad	Bệnh động mạch vành
bu	Ure máu	appet	Tình trạng ăn uống
pe	Phù chân	ane	Thiếu máu

Biến mục tiêu : classification Phân loại

Tổng quan về dataset

Sau các bước xử lý cơ bản như chỉnh Dtype , sửa lỗi điền thiếu category ta có thống kê cơ bản như sau

Numeric columns

	Tuổi	Huyết Áp	Tỷ trọng nước tiểu	Hàm lượng albumin trong nước tiểu	Mức đường trong nước tiểu	Đường huyết ngẫu nhiên	Ure máu	Creatinine huyết thanh	Natri	kali	Huyết sắc tố	Thể tích hồng cầu đóng gói
count	391.000000	388.000000	353.000000	354.000000	351.000000	356.000000	381.000000	383.000000	313.000000	312.000000	348.000000	329.000000
mean	51.483376	76.469072	1.017408	1.016949	0.450142	148.036517	57.425722	3.072454	137.528754	4.627244	12.526437	38.884498
std	17.169714	13.683637	0.005717	1.352679	1.099191	79.281714	50.503006	5.741126	10.408752	3.193904	2.912587	8.990105
min	2.000000	50.000000	1.005000	0.000000	0.000000	22.000000	1.500000	0.400000	4.500000	2.500000	3.100000	9.000000
25%	42.000000	70.000000	1.010000	0.000000	0.000000	99.000000	27.000000	0.900000	135.000000	3.800000	10.300000	32.000000
50%	55.000000	80.000000	1.020000	0.000000	0.000000	121.000000	42.000000	1.300000	138.000000	4.400000	12.650000	40.000000
75%	64.500000	80.000000	1.020000	2.000000	0.000000	163.000000	66.000000	2.800000	142.000000	4.900000	15.000000	45.000000
max	90.000000	180.000000	1.025000	5.000000	5.000000	490.000000	391.000000	76.000000	163.000000	47.000000	17.800000	54.000000

Tổng quan về dataset

Sau các bước xử lý cơ bản như chỉnh Dtype , sửa lỗi điền thiếu category ta có thống kê cơ bản như sau

Category columns

	Tình trạng hồng cầu	Tình trạng bạch cầu mủ	Cụm bạch cầu mủ	Vị khuẩn trong nước tiểu	Tăng huyết áp	Tiểu đường	Bệnh động mạch vành	Tình trạng ăn uống	Phù chân	Thiếu máu	Phân loại
count	248	335	396	396	398	398	398	399	399	399	400
unique	2	2	2	2	2	2	2	2	2	2	2
top	normal	normal	notpresent	notpresent	no	no	no	good	no	no	ckd
freq	201	259	354	374	251	261	364	317	323	339	250



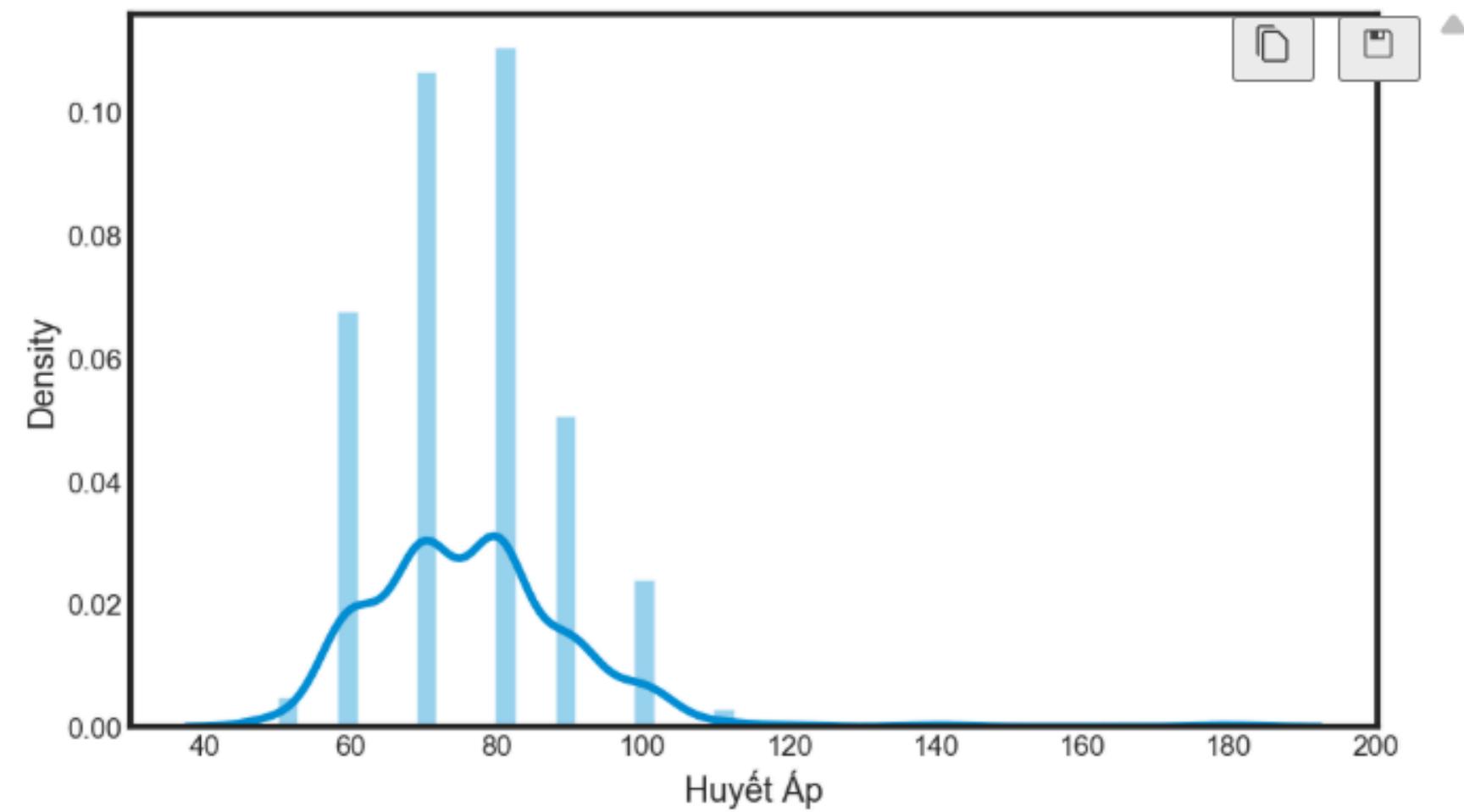
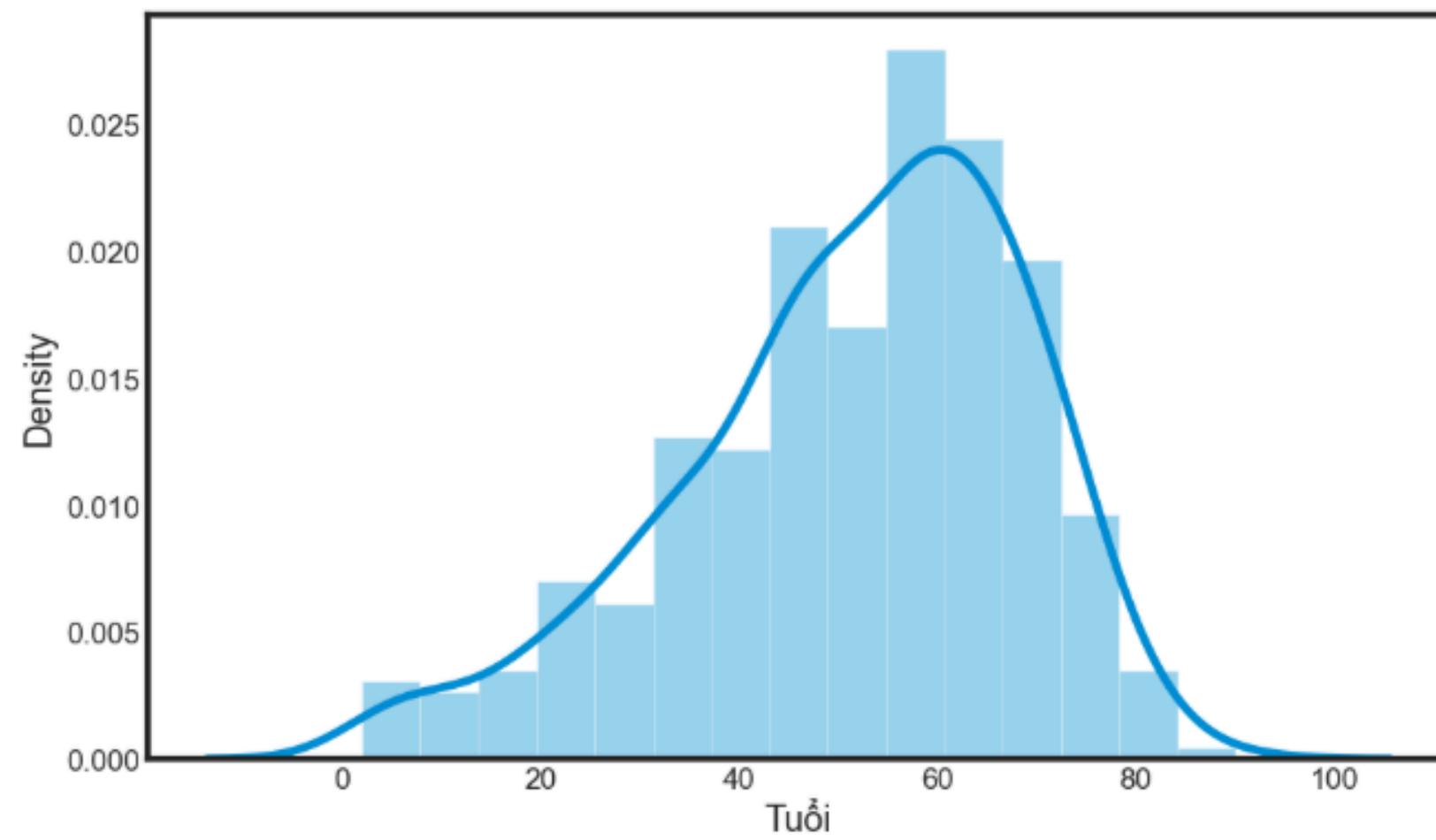
Exploratory Data Analysis (EDA)

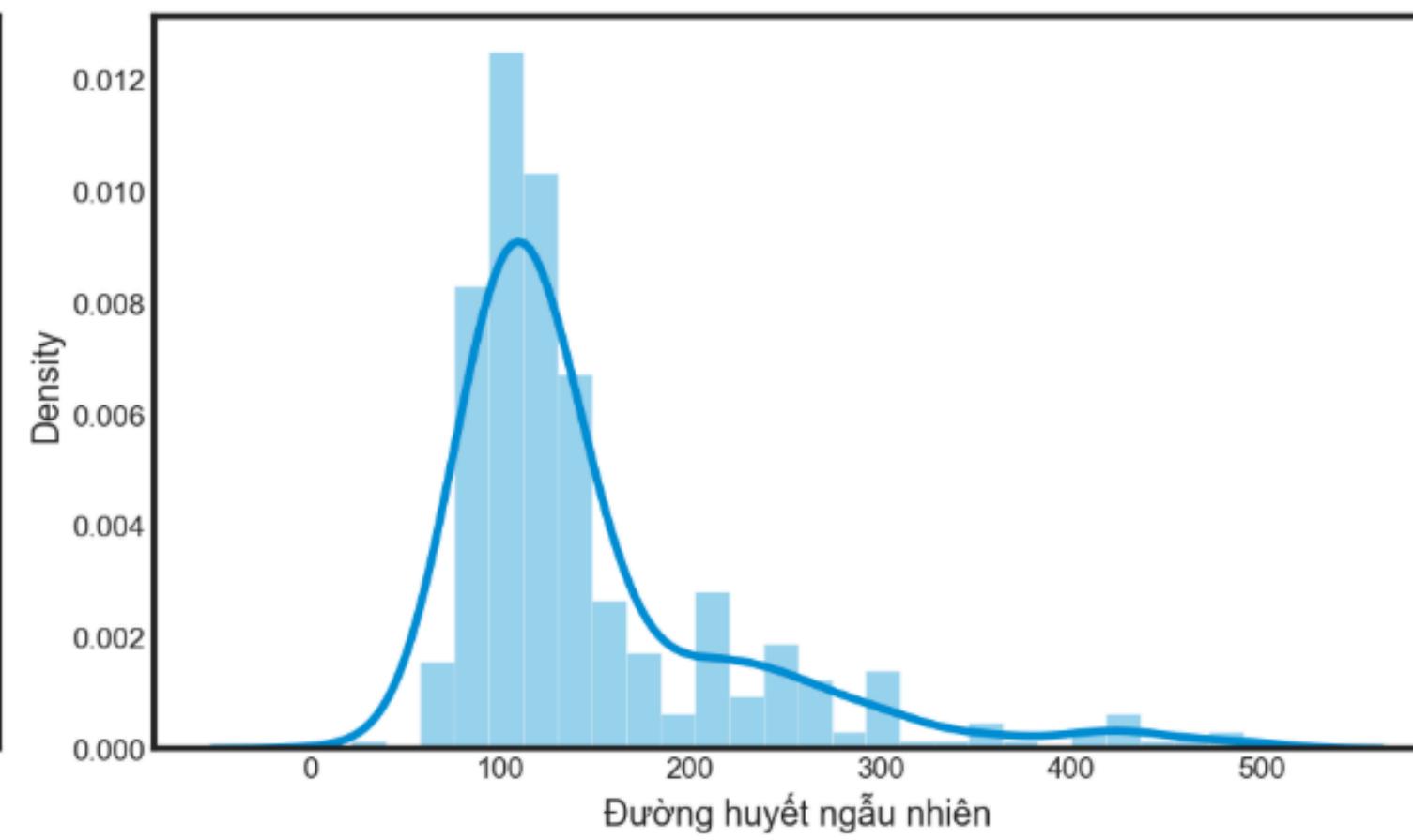
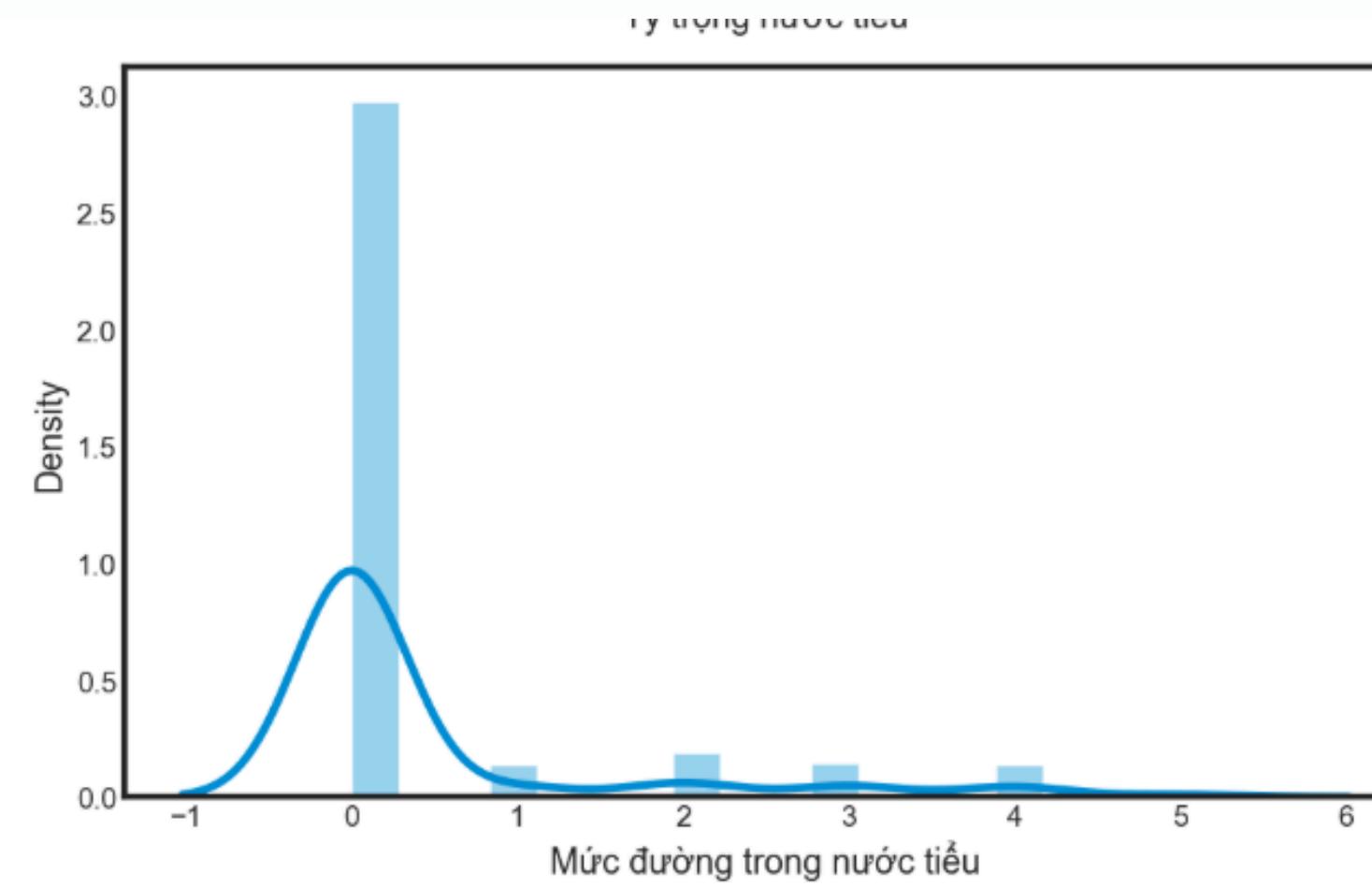
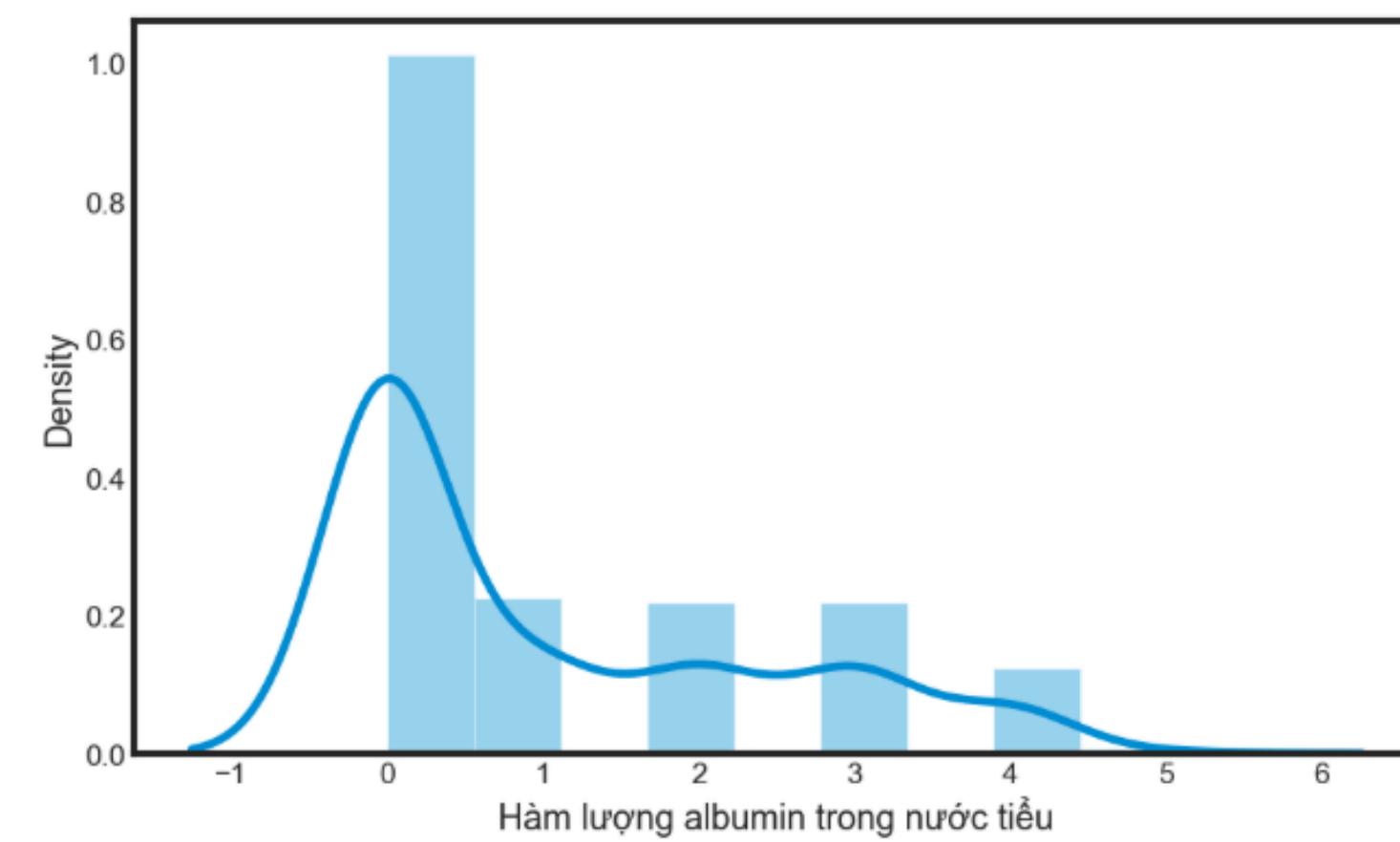
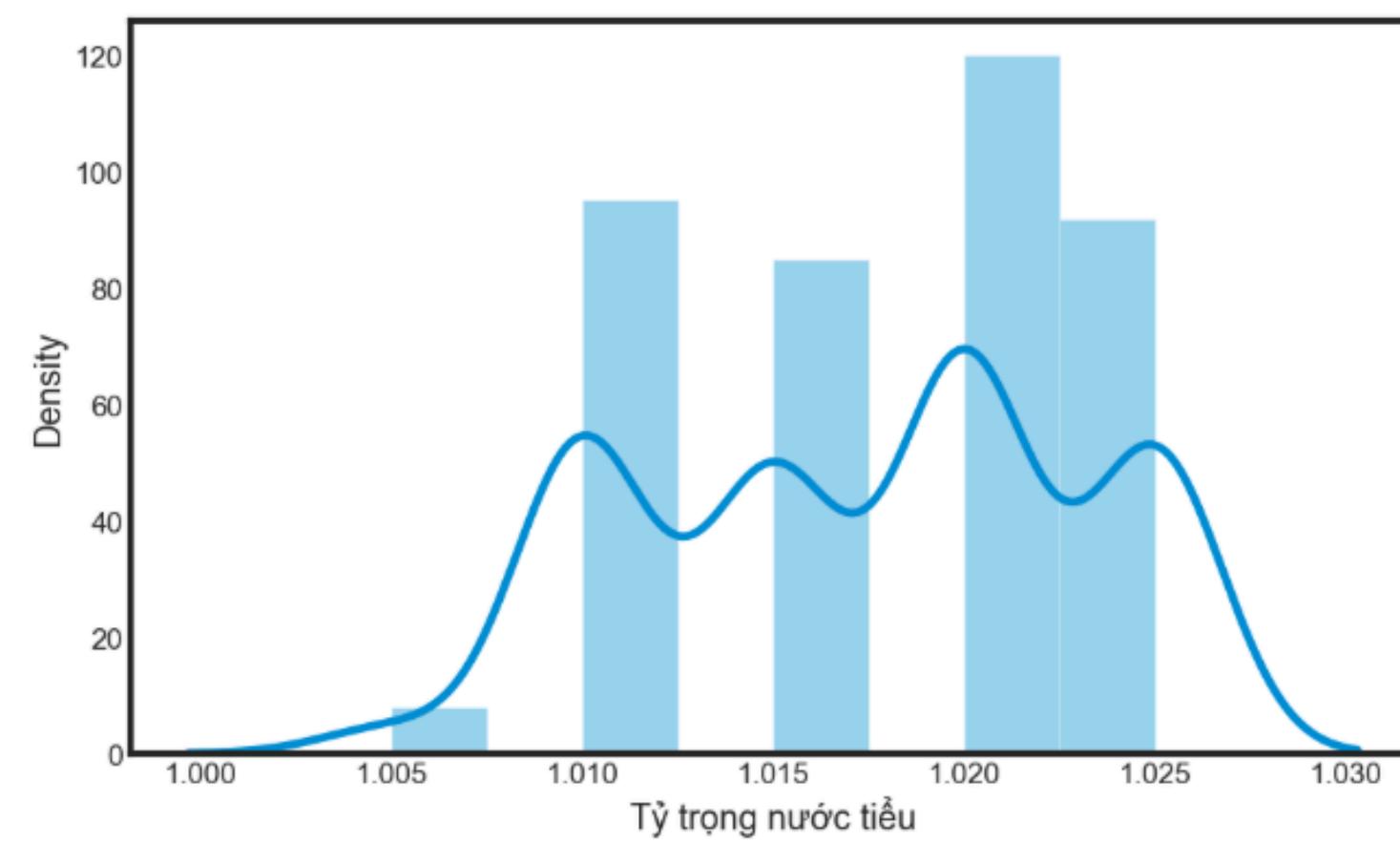
EDA là một bước quan trọng trong khoa học dữ liệu và phân tích dữ liệu, vì nó giúp trực quan hóa dữ liệu để hiểu các đặc điểm chính, phát hiện các mẫu (patterns) và khám phá cách các thành phần khác nhau của dữ liệu liên kết với nhau.



Kiểm tra phân phối

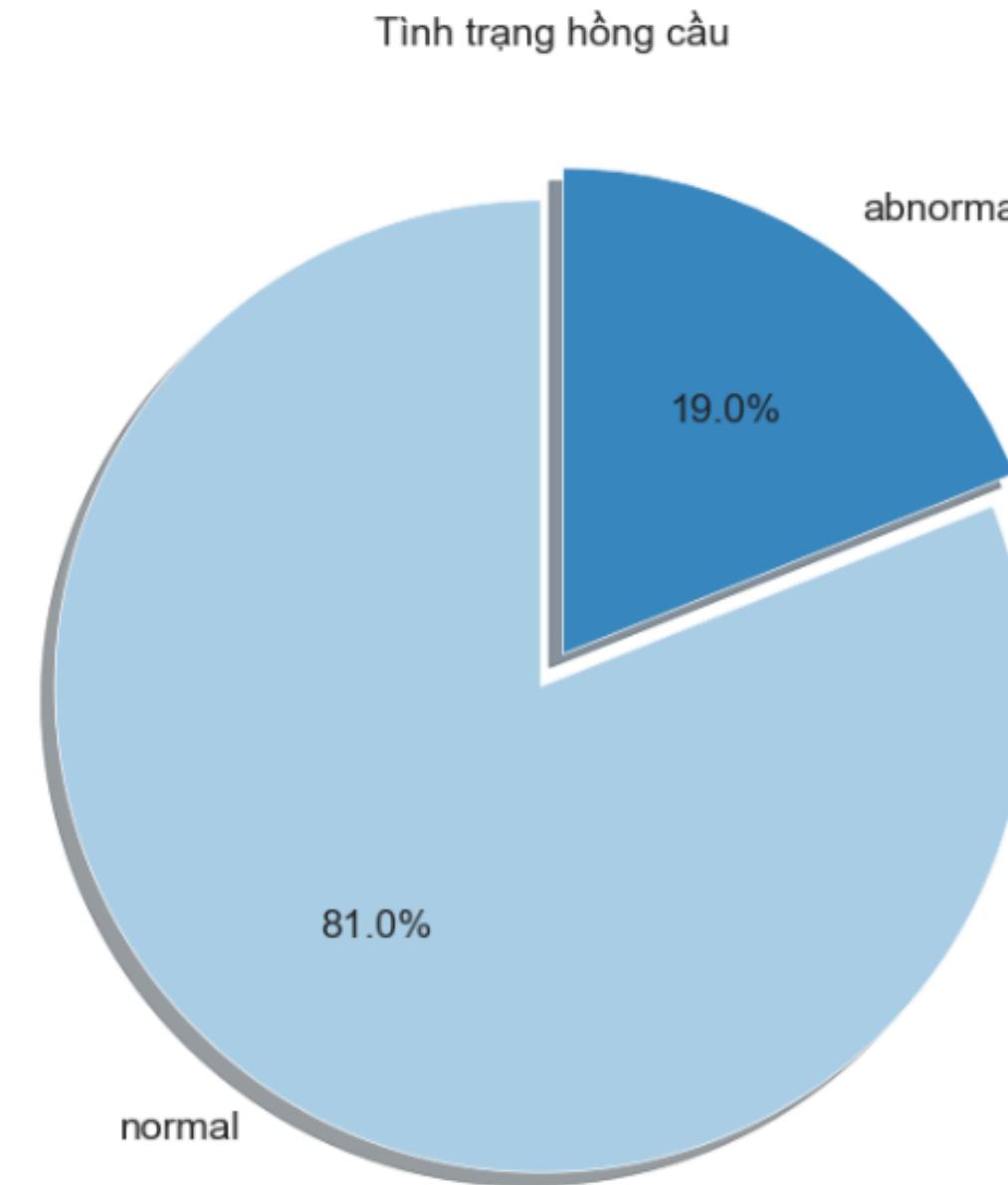
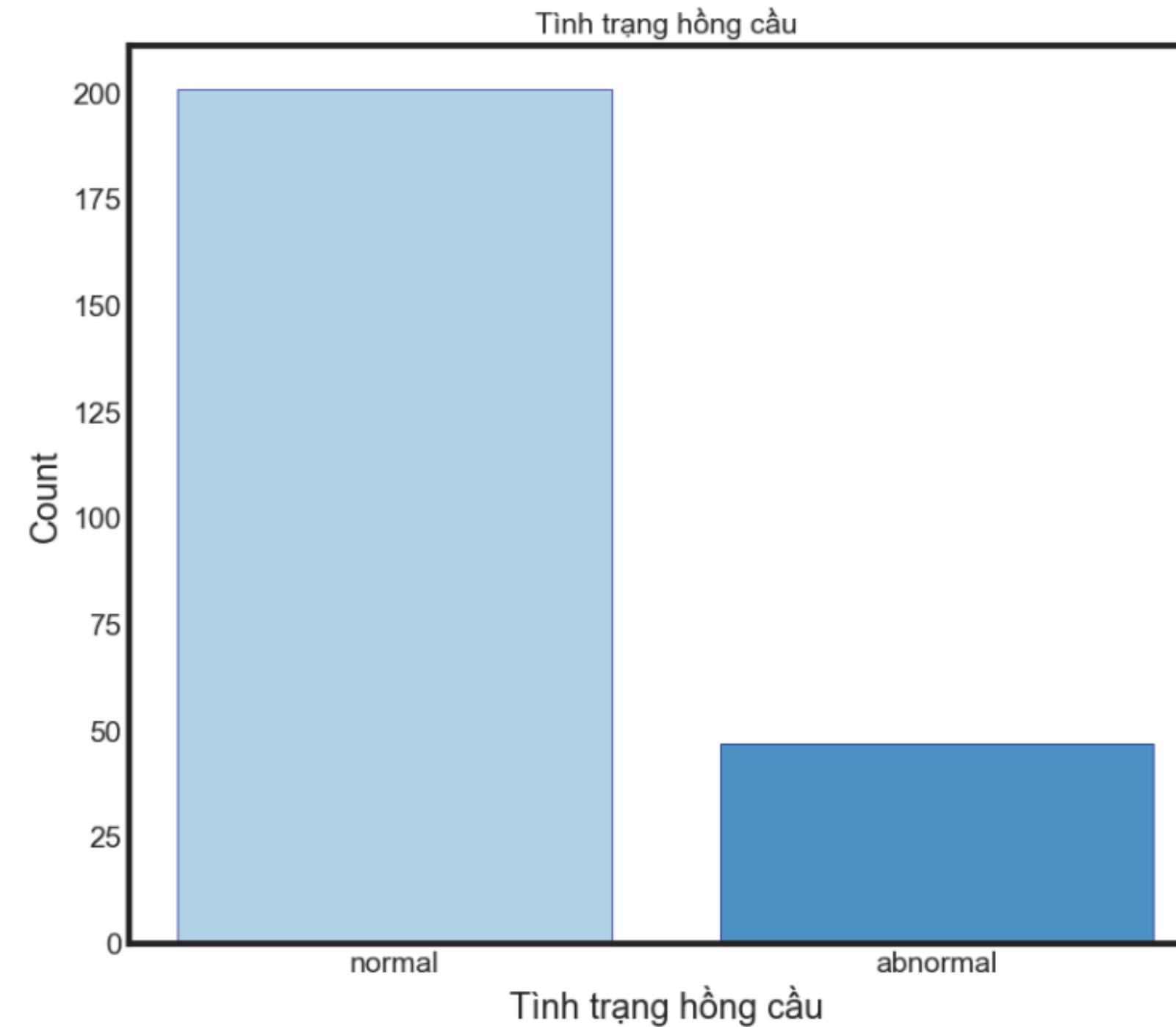
Numeric columns

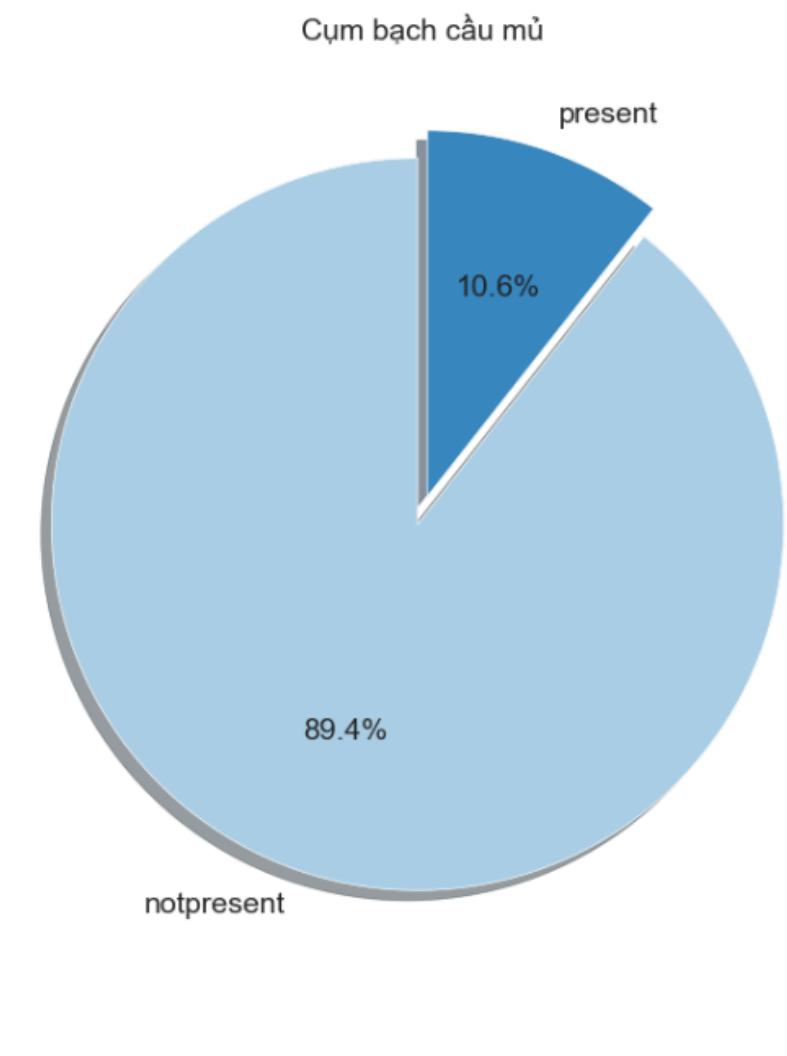
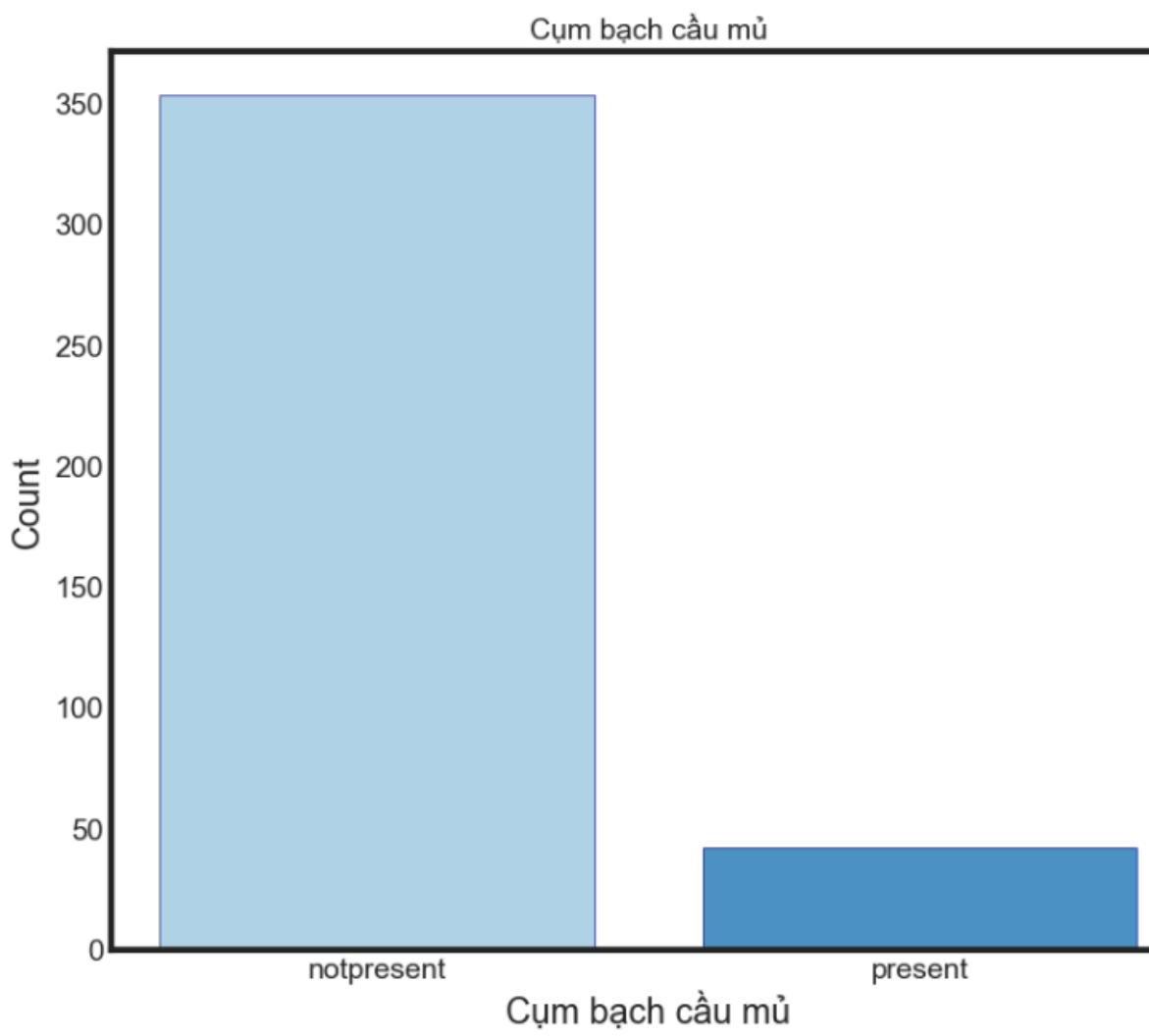
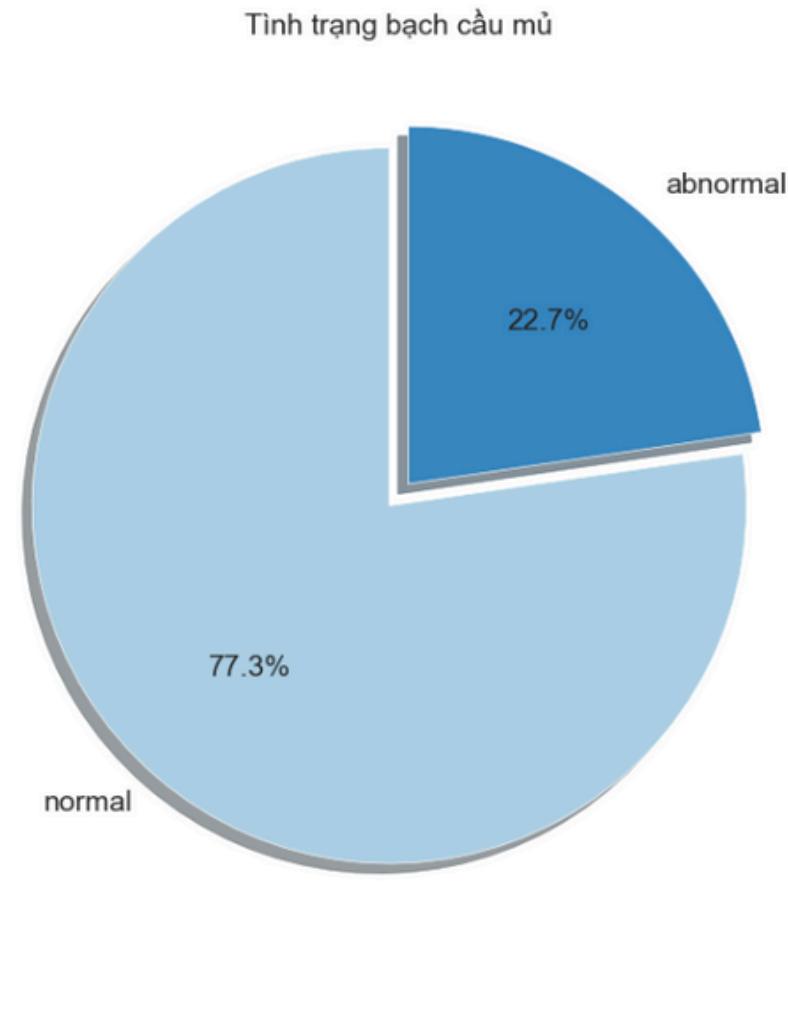
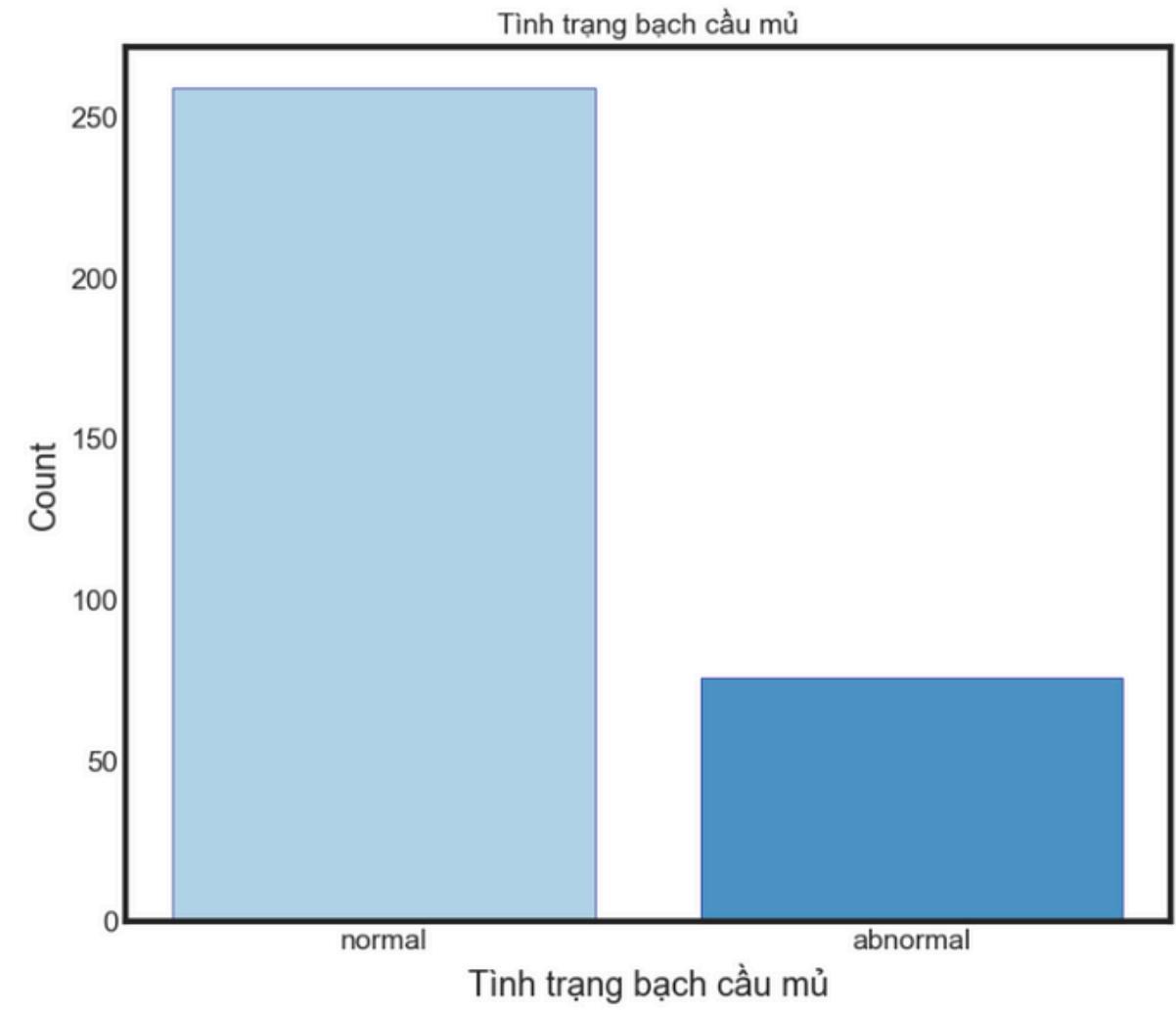




Kiểm tra phân phối

Category columns





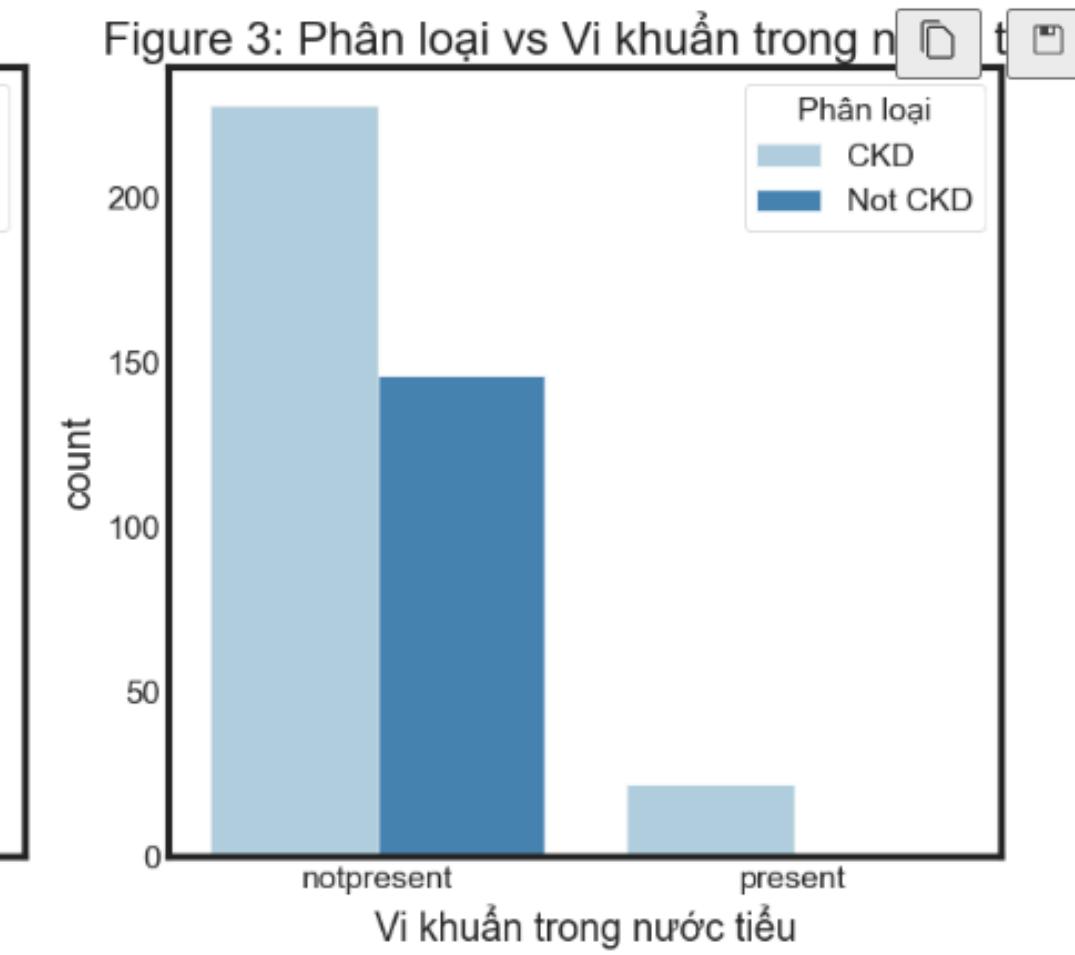
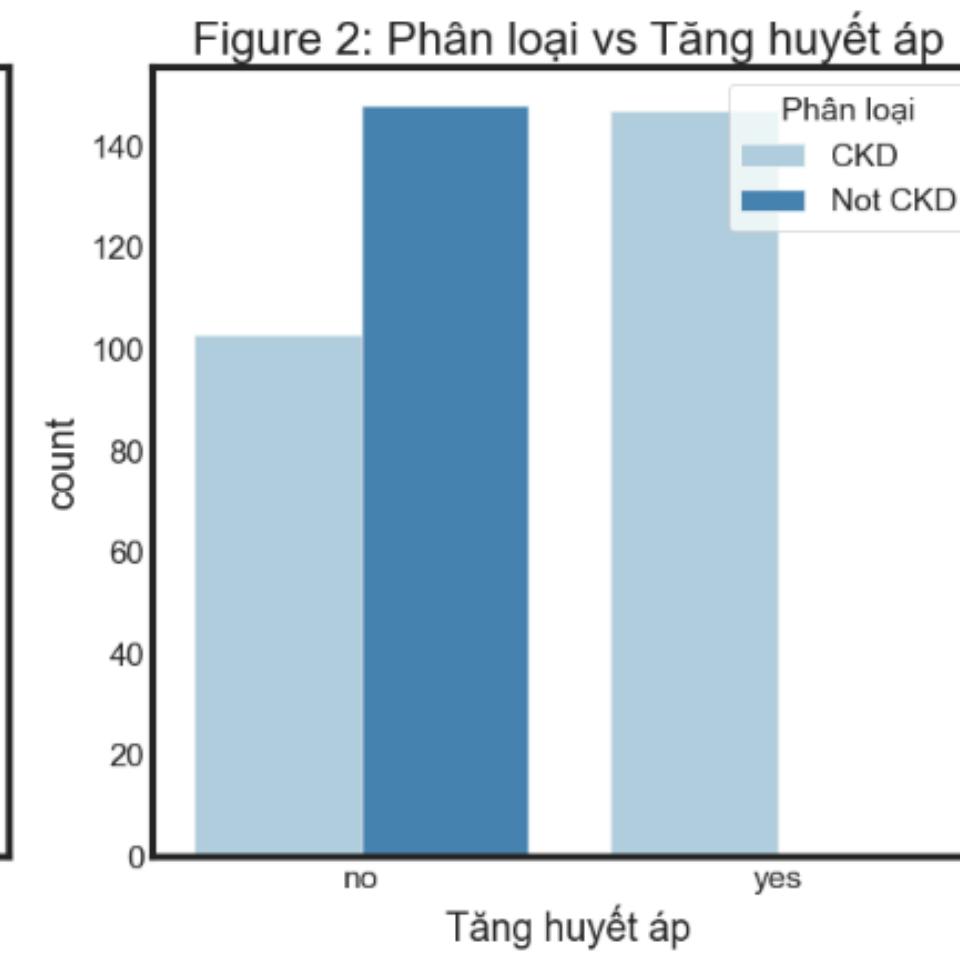
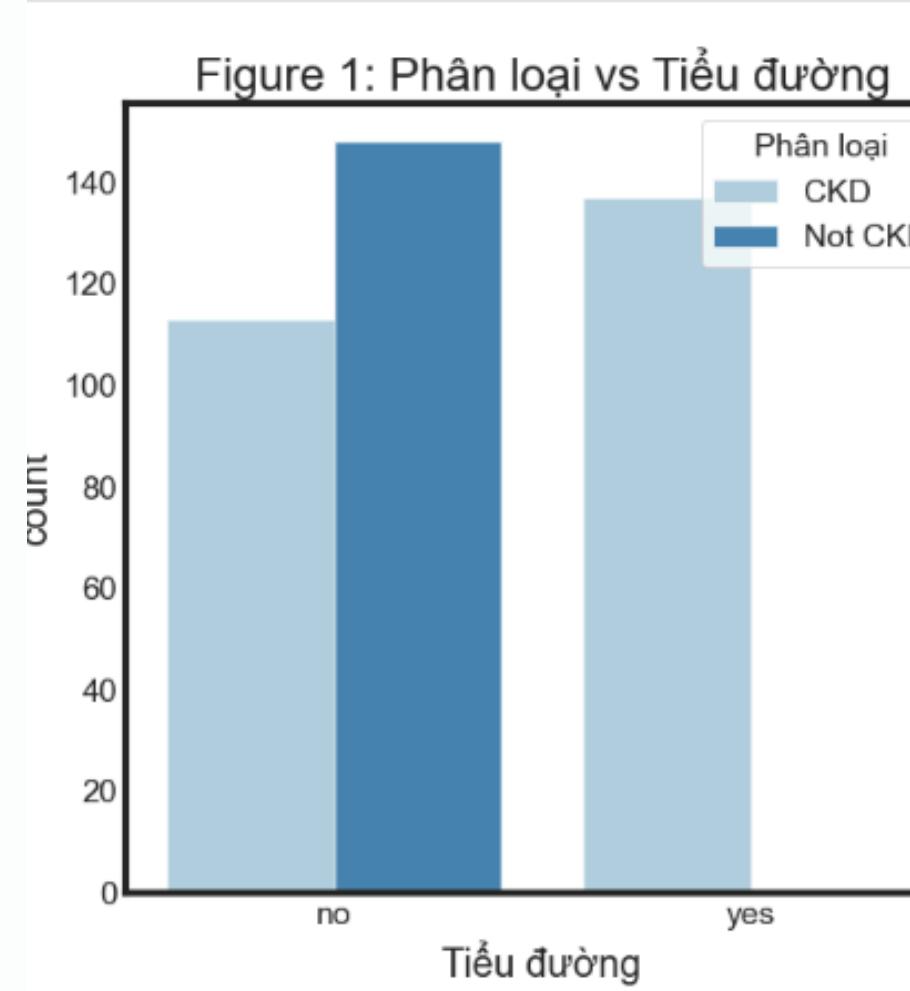
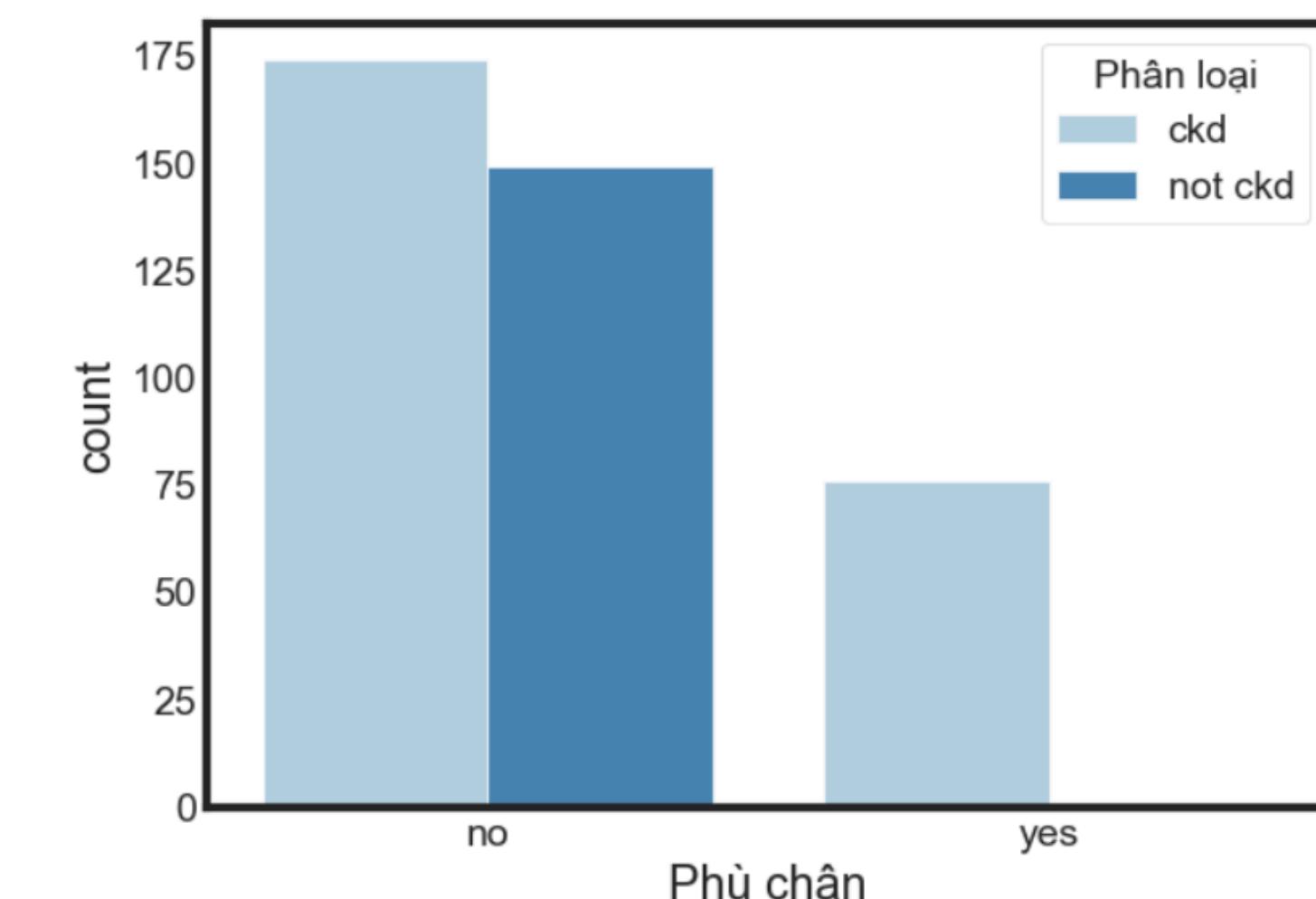
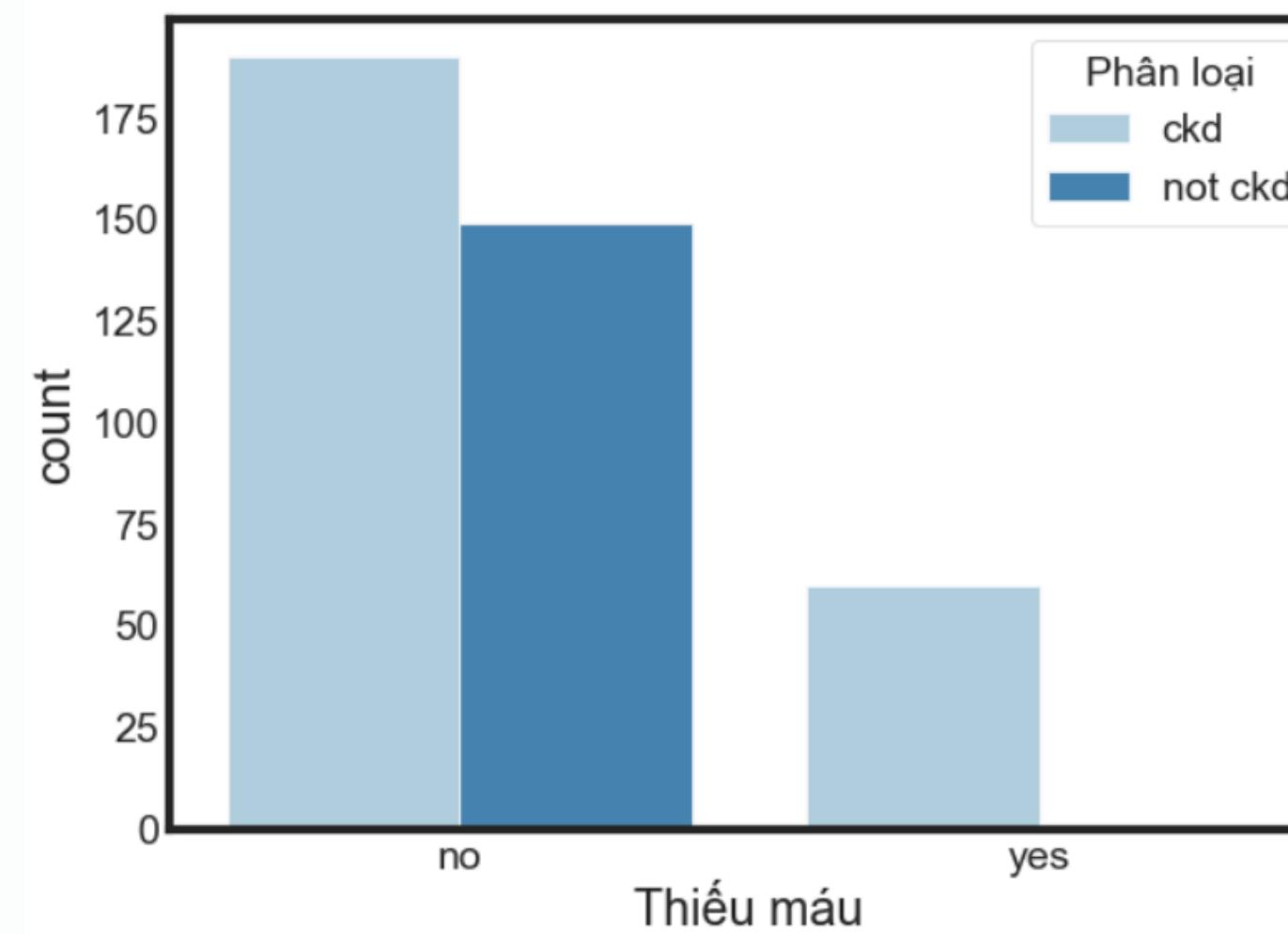
Tương quan đặc trưng category với target

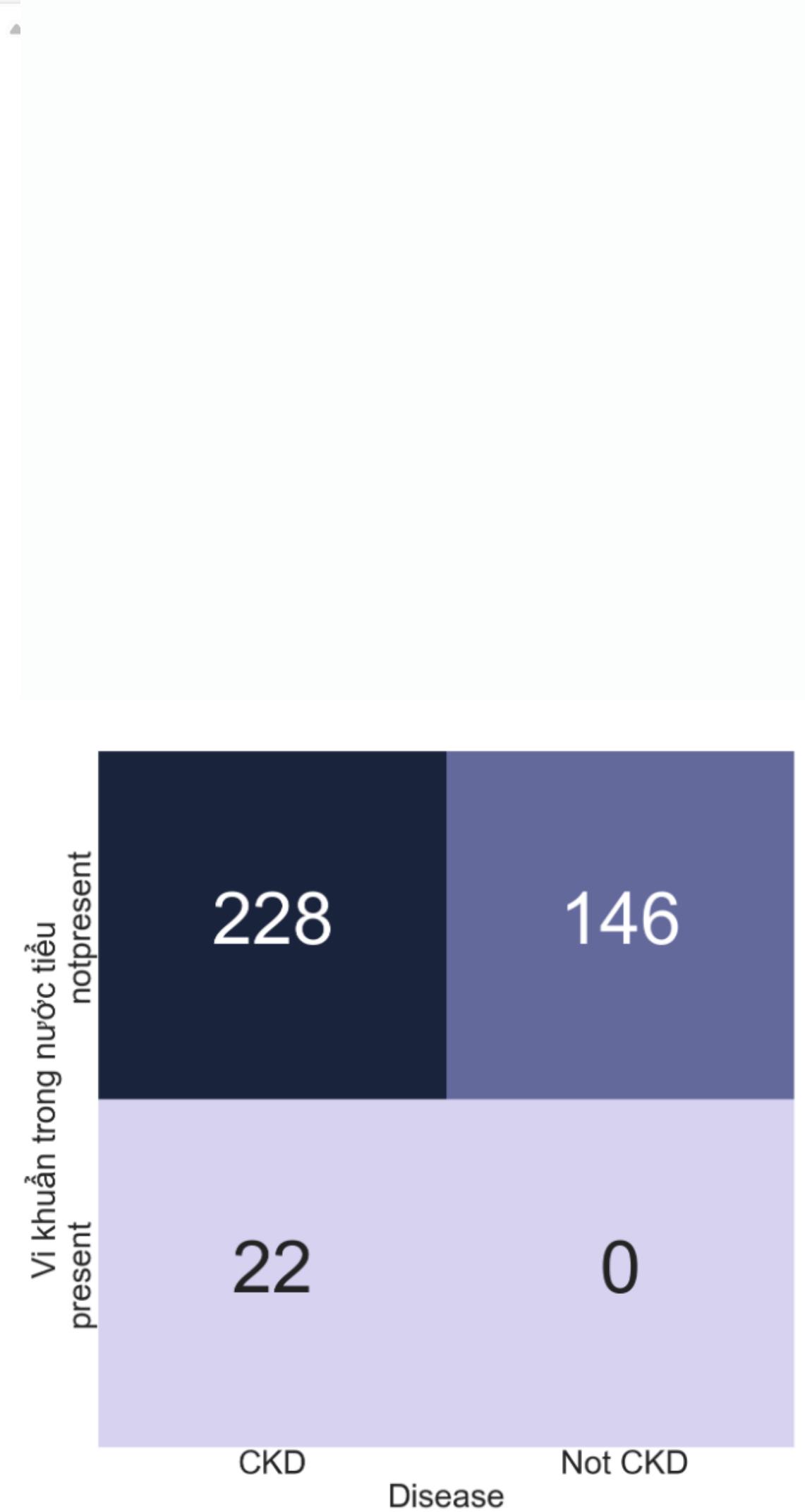
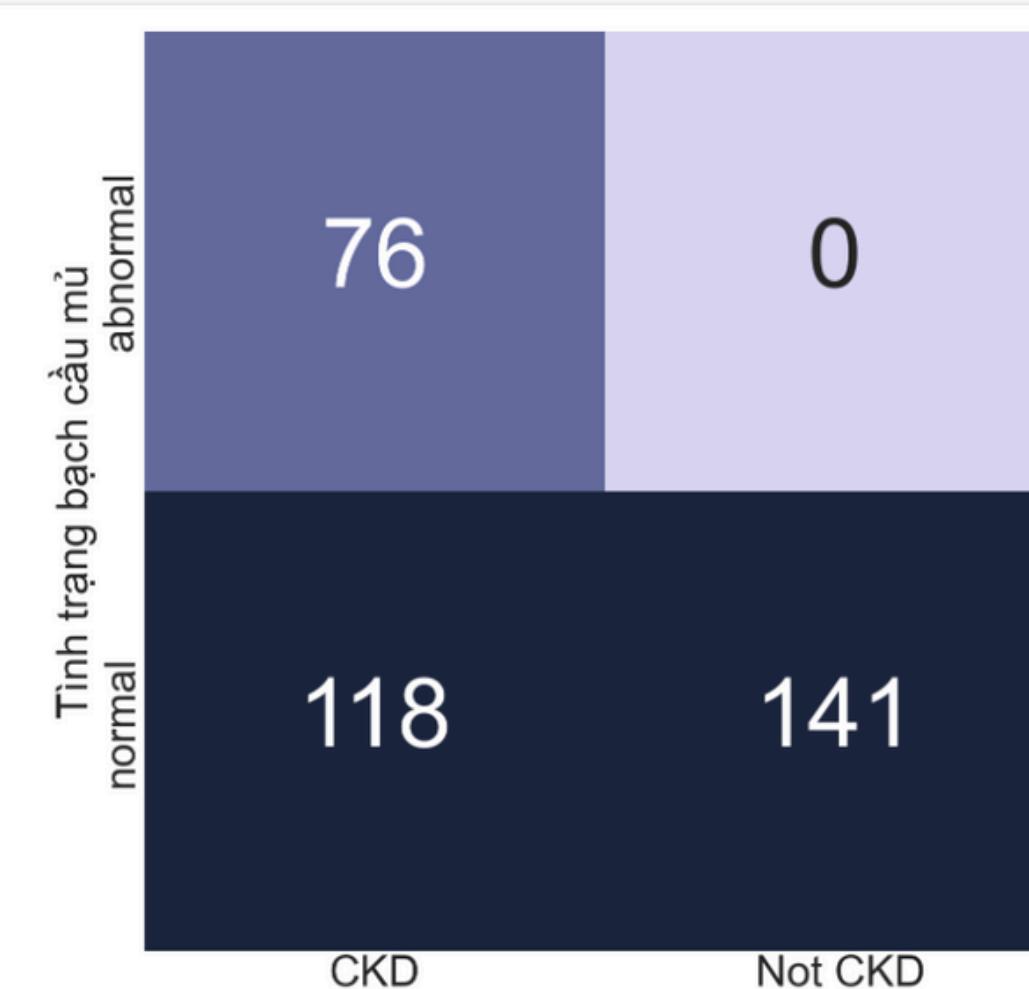
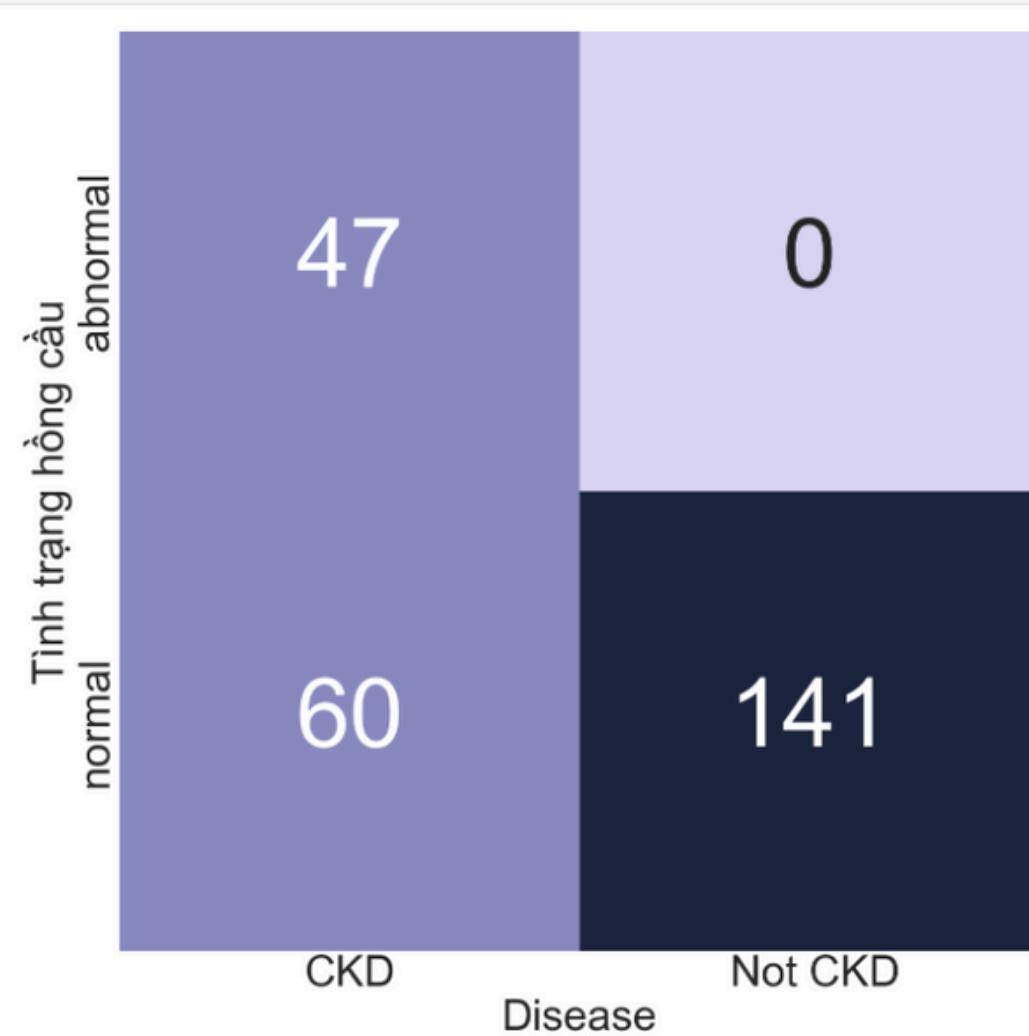
Category columns

Vấn đề cân bằng dữ liệu của target

```
✓ df["Phân loại"].value_counts(normalize=True).to_frame()*100
]
   proportion
Phân loại
  ckd      62.5
not ckd     37.5
```

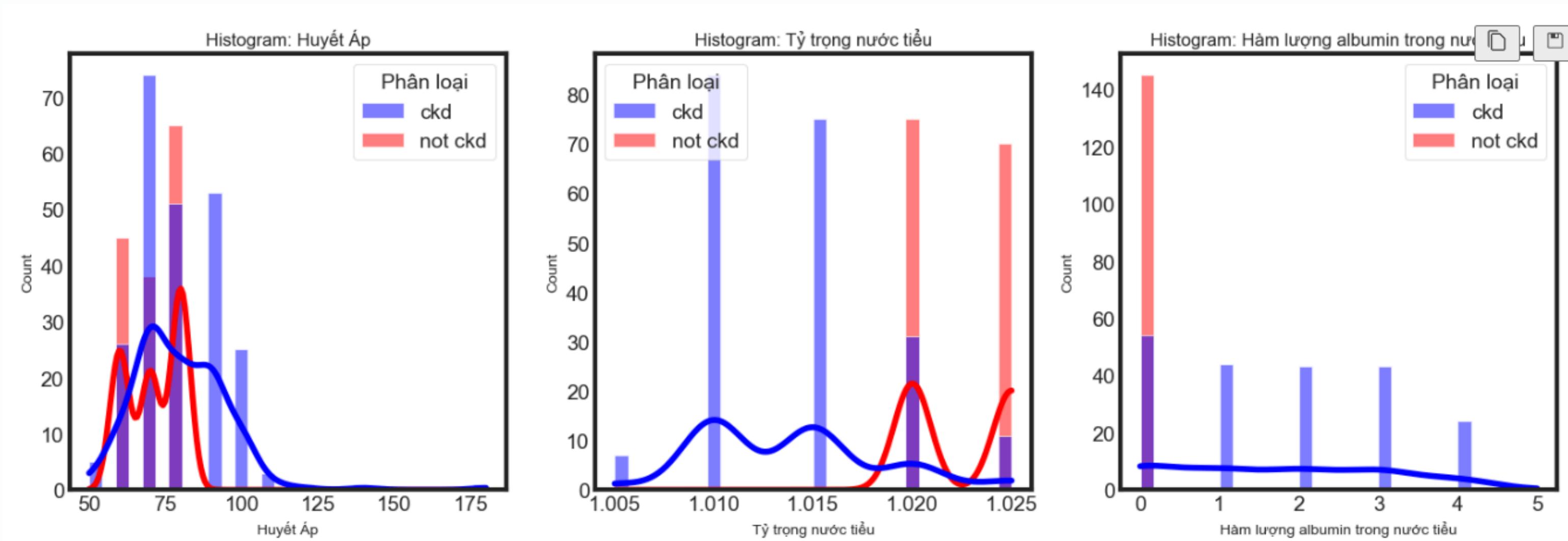
Target bị mất cân bằng dữ liệu những không quá nghiêm trọng

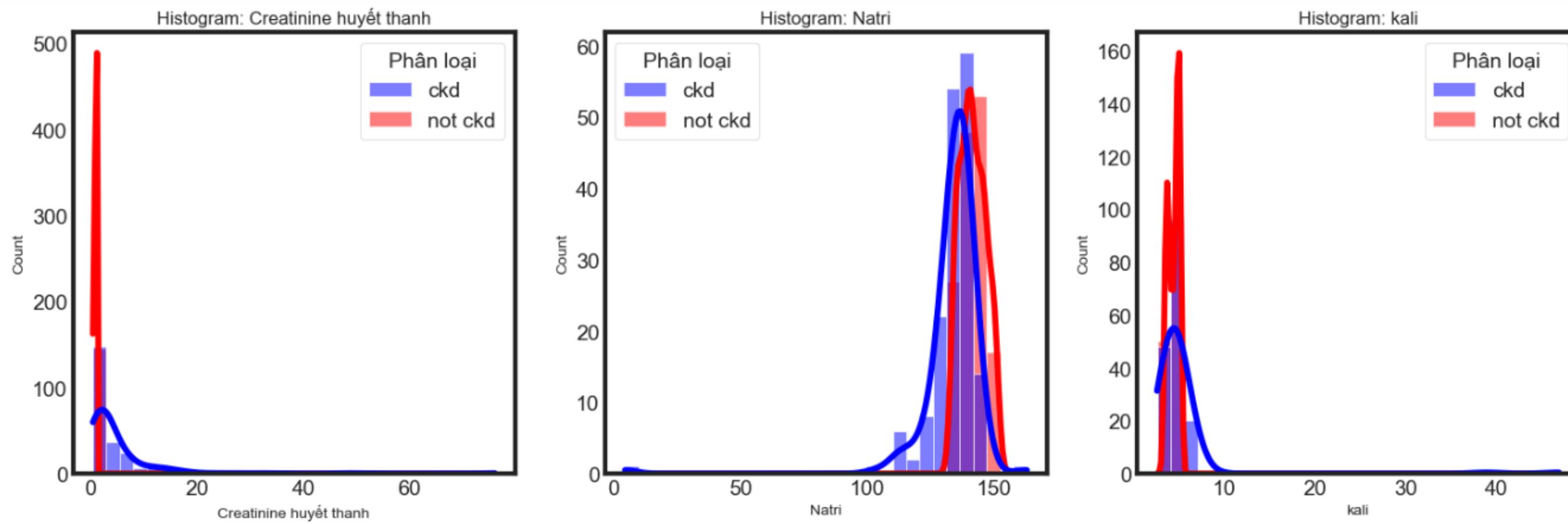
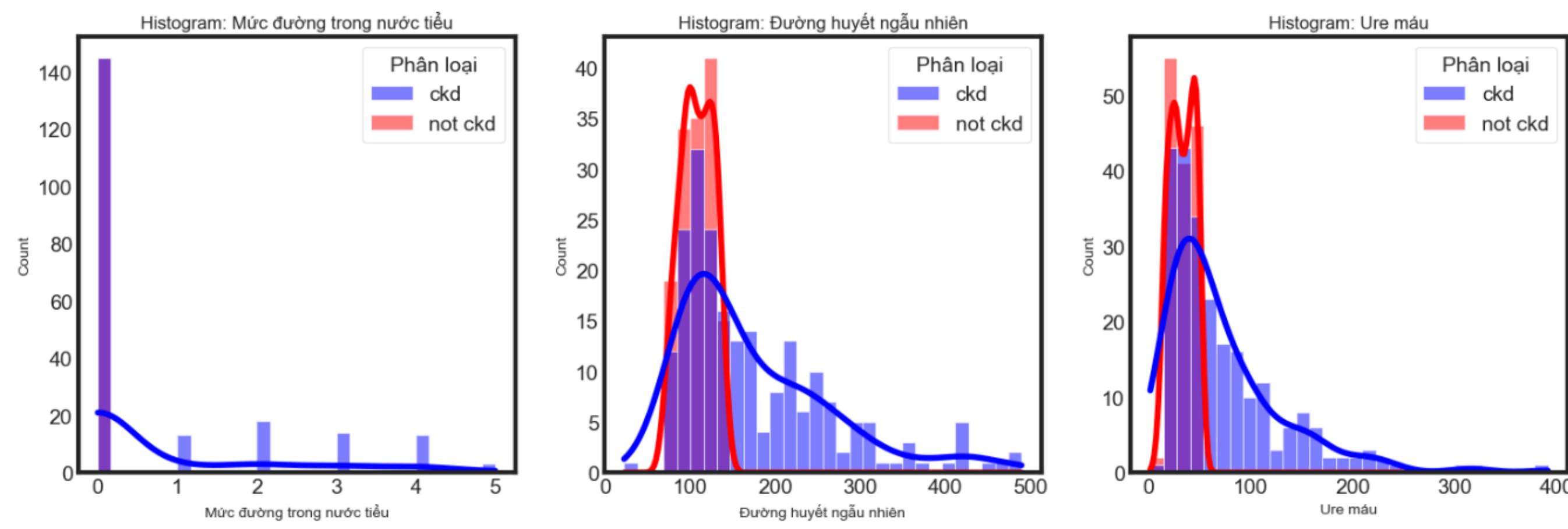


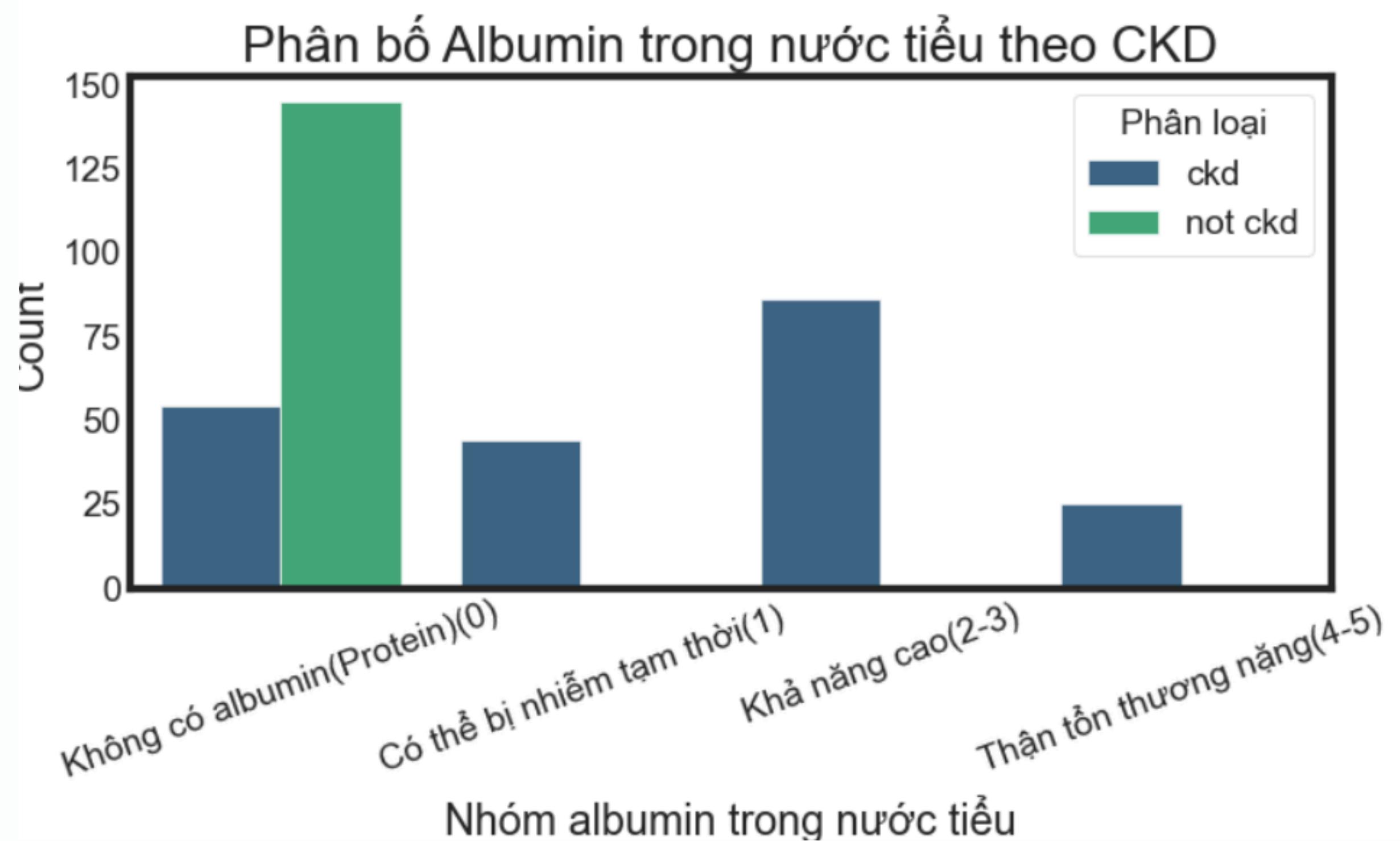


Tương quan đặc trưng Numeric với Target

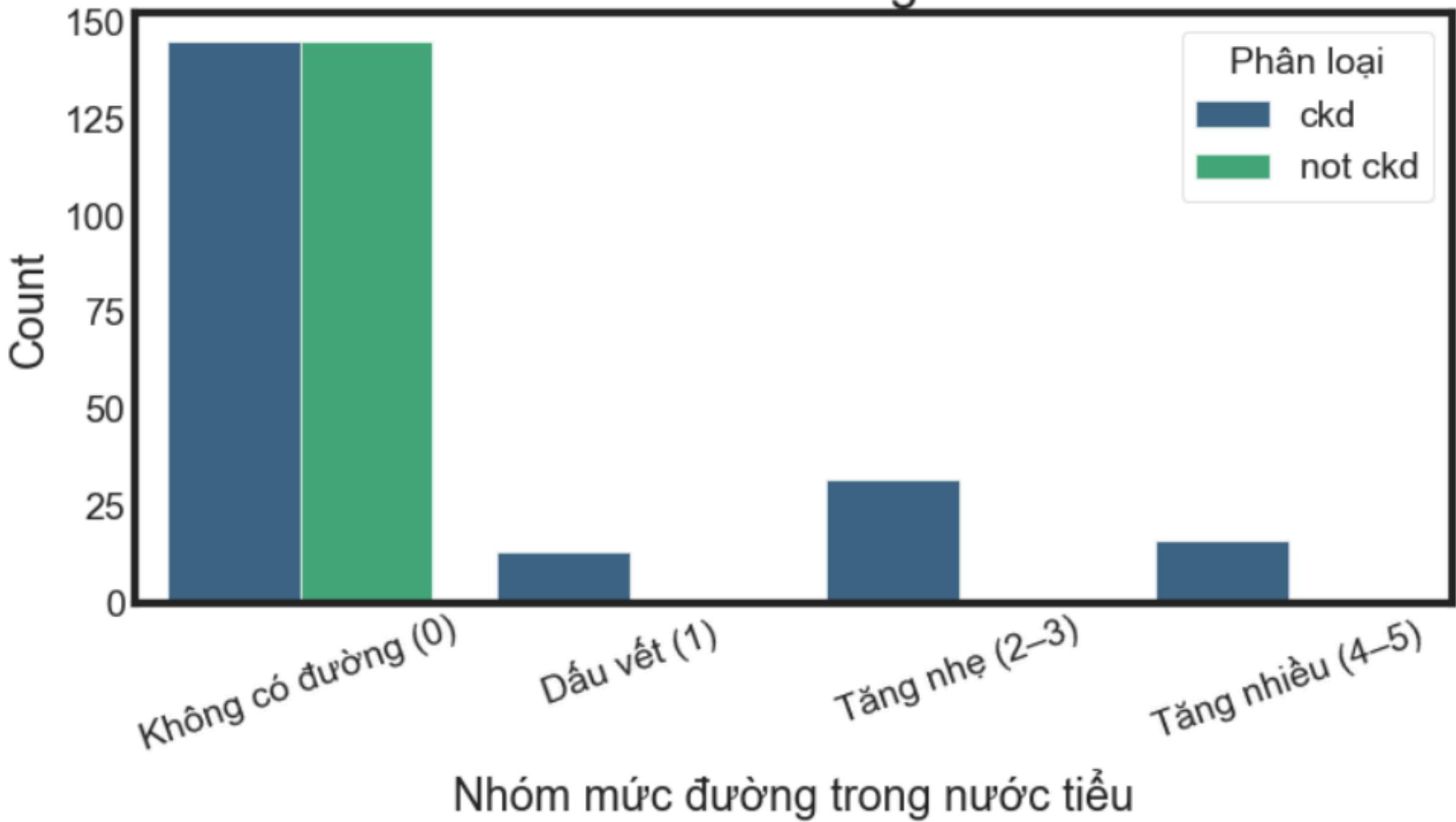
Numeric columns



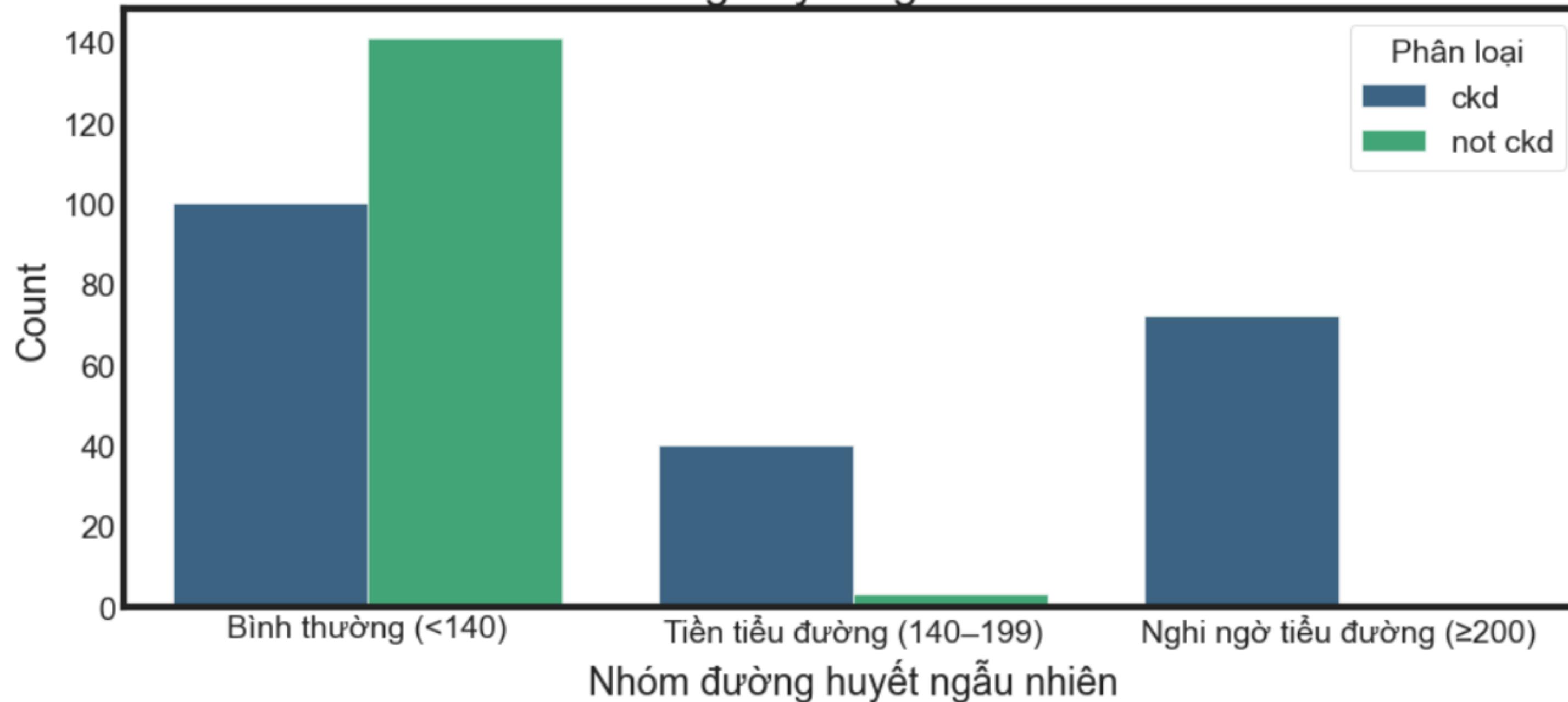


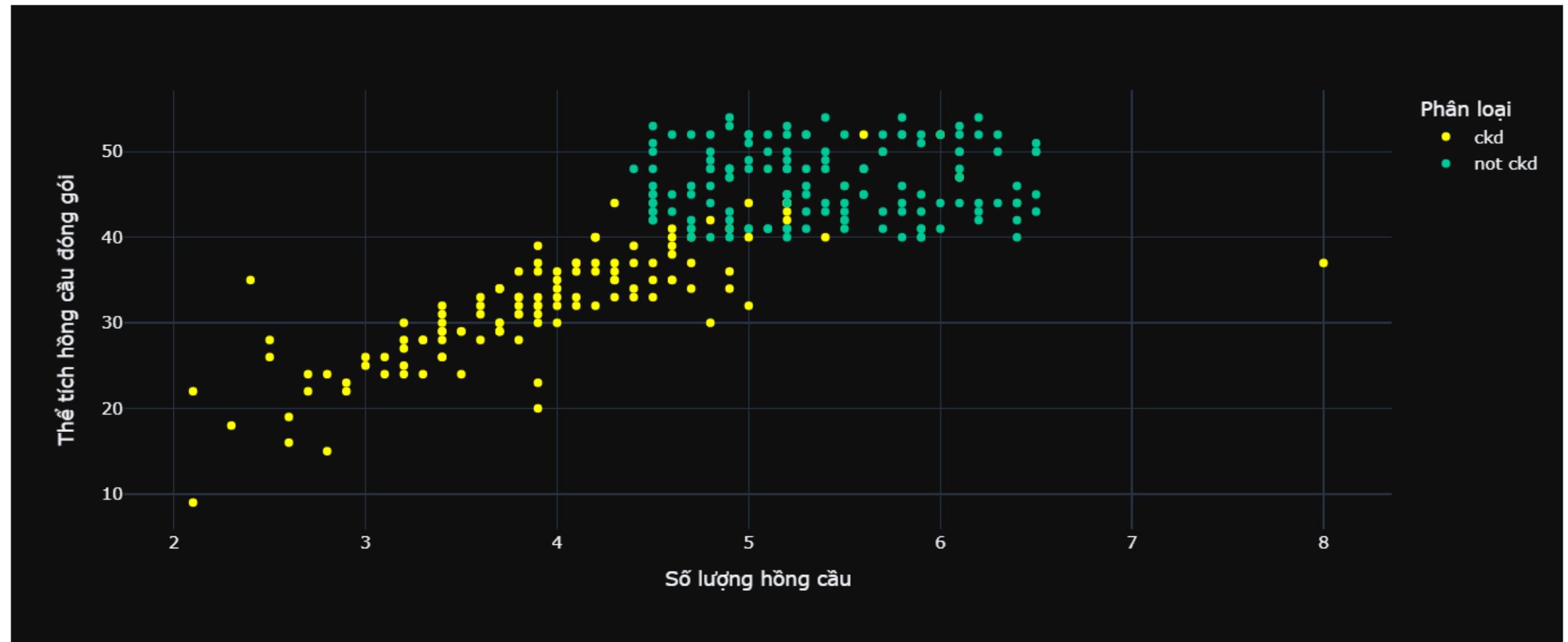
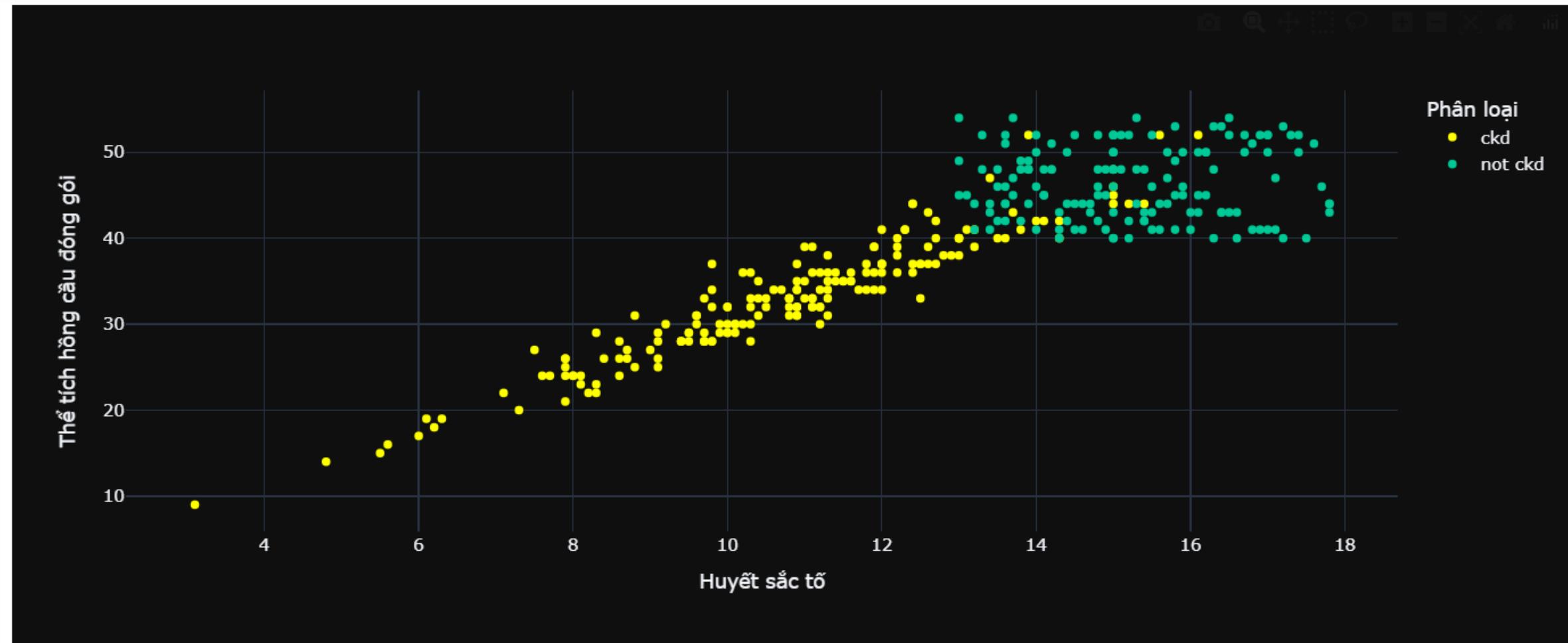


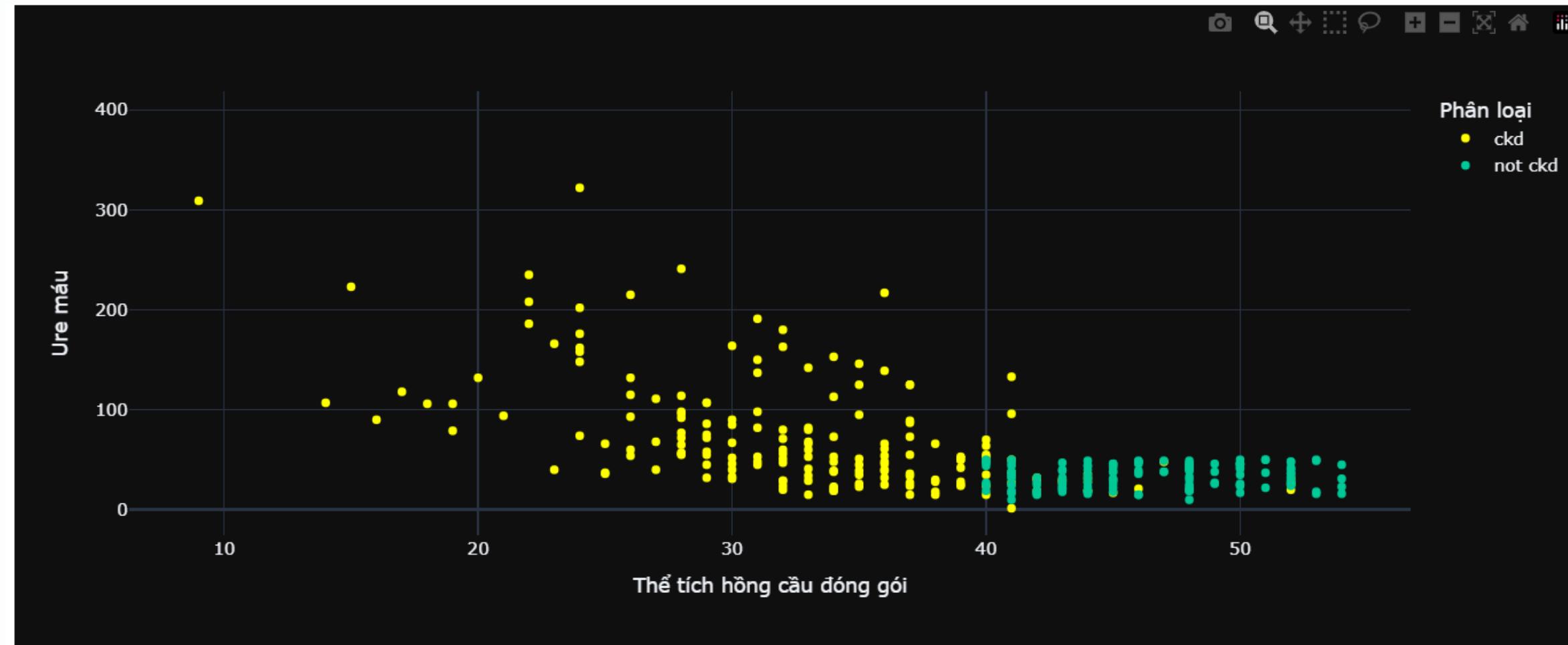
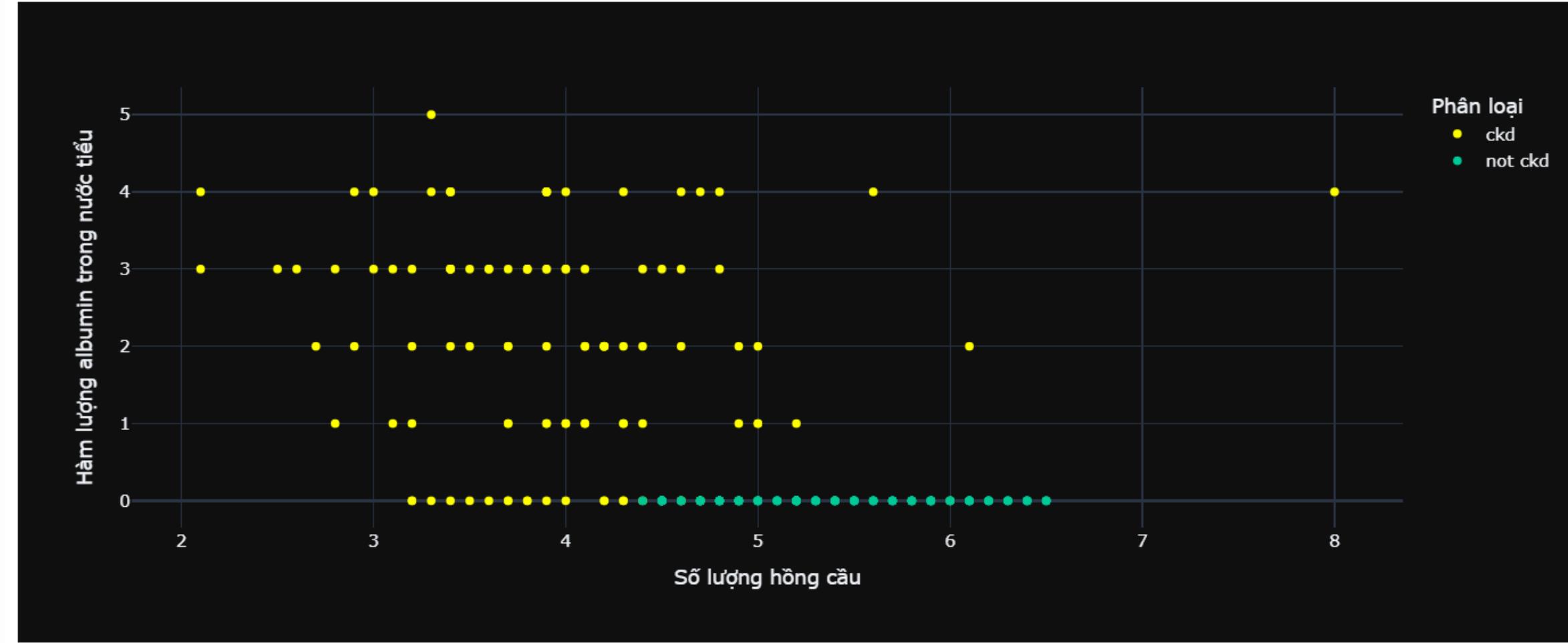
Phân bố Mức đường theo CKD



Phân bố Đường huyết ngẫu nhiên theo CKD







Kết luận sau khi EDA

DataPreprocessing

Tiền xử lý dữ liệu là bước đầu tiên trong quy trình phân tích dữ liệu, trong đó dữ liệu thô được làm sạch, tổ chức và biến đổi thành định dạng phù hợp cho các bước phân tích tiếp theo. Quá trình này bao gồm một loạt các thao tác nhằm nâng cao chất lượng dữ liệu và chuẩn bị dữ liệu cho các mô hình học máy và mô hình thống kê.



Nhắc lại mục tiêu

Sau rất nhiều lần thử chúng tôi nhận thấy rằng đây là một dataset đặc biệt và căn bệnh này cũng rất đặc biệt

Thứ nhất căn bệnh suy thận là một bệnh mà cực kì khó phát hiện khi còn ở giai đoạn sớm tức từ giao đoạn 2 trở về

Thứ 2 Những triệu chứng và chỉ số sinh hóa của người mà mắc bệnh từ giai đoạn 2b trở lên thì cực kì khác biệt với những người khỏe mạnh . Vì thận là một trong những cơ quan quan trọng bậc nhất trong cơ thể nên nếu nó gặp vấn đề thì chỉ số giữa người bệnh và người không mắc cực kì khác biệt

Nhưng có vài chỉ số chỉ khác biệt với người không mắc khi đã ở những giai đoạn muộn và khi ở giai đoạn muộn thì không thể phục hồi

Thông tin được lấy từ các trang báo và có thể thấy việc đó thông qua EDA mà chúng tôi đã làm

Nhắc lại mục tiêu

Vậy câu hỏi đặt ra cho chúng tôi lúc này là : Nếu chỉ số sinh hóa và triệu chứng của người bị bệnh và người không bị khác nhau và tách biệt như vậy thì cần gì mô hình dự đoán nữa. Bởi vì chỉ cần nhìn vào vài chỉ số là có thể phân biệt được mà . Và đây cũng là câu hỏi khiến chúng tôi băn khoăn

Trước hết chúng ta cần phải hiểu quy trình đi khám đặc biệt là xét nghiệm với những cơ quan quan trọng trong cơ thể
Cái này có thể tự tìm hiểu

Với số lượng bệnh nhân ngày càng trẻ hóa và lớn như hiện nay thì các bác sĩ phải làm việc với khối lượng công việc lớn dẫn đến có thể có sai sót

Nhắc lại mục tiêu

Vậy nên để giải quyết cho vấn đề đó chúng tôi có ý tưởng như sau :

Chúng tôi sẽ làm 2 mô hình

Model đầu tiên sẽ ứng dụng trong việc rà soát và dự đoán sớm bệnh :
lý do để làm việc này đã được đề cập phía trên

Model thứ 2 sẽ dùng để xác nhận rằng người đó bị bệnh :
ý do để làm việc này đã được đề cập phía trên

Ý tưởng của model dự đoán sớm

Như đã đề cập và thông qua bước EDA chúng ta nhận thấy rằng có vài chỉ số sinh hóa có thể phân biệt bệnh cực kì rõ . Và hầu như đều ở giai đoạn muộn .

Vậy nên chúng tôi sẽ lược bỏ bớt những feature mang tính muộn màng đó đi chỉ giữ lại các feature có khả năng phát hiện bệnh sớm

Để chọn được những feature đó chúng tôi dựa theo nhiều tiêu chí :
Thường được xét nghiệm trong các cuộc khám sức khỏe định kỳ
Không có cấu trúc mạnh mẽ . Ví dụ như grf < 80 => ckd
Và thông qua 1 vài trang báo và sự tham khảo .

Sau đó chúng tôi sẽ làm những bước như train và đánh giá hiệu năng nhưng bình thường

Ý tưởng của model trong việc xác nhận

Đối với model thứ 2 này chúng tôi sẽ train với toàn bộ thuộc tính

Ý nghĩa của mô hình này là : Trợ giúp cho các bác sĩ có thể xác nhận và không tốn thời gian ở những bước rườm rà khác . Cho các bác sĩ có thời gian nghỉ ngơi tĩnh táo để hoàn thành tốt công việc của mình

Sau đó chúng tôi sẽ làm những bước như train và đánh giá hiệu năng nhưng bình thường

Model dự đoán sớm

Để tránh việc tập test bị rò rỉ ở train chúng tôi sẽ chia train test trước sau đó mới đến việc fill miss , mã hóa , chuẩn hóa

```
# Chia X và Y
ind_col = ['Tuổi',
'Huyết Áp',
'Tỷ trọng nước tiểu',
'Hàm lượng albumin trong nước tiểu',
'Mức đường trong nước tiểu',
'Tình trạng hồng cầu',
'Tình trạng bạch cầu mủ',
'Tăng huyết áp',
'Tiểu đường',
'Tình trạng ăn uống']
dep_col = 'Phân loại'

x = df[ind_col]
y = df[dep_col]
```

Python

```
# Chia train và test
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_state = 42,stratify=y)
```

Python

Model dự đoán sớm

Vì dữ liệu thiếu ở các cột không quá 70% nên vẫn giữ lại

	missing_data	missing_data_ratio
Tuổi	9	2.2
Huyết Áp	12	3.0
Tỷ trọng nước tiểu	47	11.7
Hàm lượng albumin trong nước tiểu	46	11.5
LAI\Hamluongalbumin(Protein)trongnuoctieu.ipynb	49	12.2
Tình trạng hồng cầu	152	38.0
Tình trạng bạch cầu mủ	65	16.2
Cụm bạch cầu mủ	4	1.0
Vi khuẩn trong nước tiểu	4	1.0
Đường huyết ngẫu nhiên	44	11.0
Ure máu	19	4.7
Creatinine huyết thanh	17	4.2
Natri	87	21.7
kali	88	22.0
Huyết sắc tố	52	13.0
Thể tích hồng cầu đóng gói	71	17.7
Số lượng bạch cầu	106	26.5
Số lượng hồng cầu	131	32.7
Tăng huyết áp	2	0.5
Tiểu đường	2	0.5
Bệnh động mạch vành	2	0.5
Tình trạng ăn uống	1	0.2

Model dự đoán sớm

Trong đây chúng tôi điền giá trị bị thiếu với nhóm numeric bằng trung vị và nhóm category bằng giá trị xuất hiện nhiều nhất . Mục đích tôi sẽ giải thích trên bảng

```
# Xử lý missing với các cột numeric cho X_Train và X_Test
from sklearn.impute import SimpleImputer
num_cols = ['Tuổi',
'Huyết Áp',
'Tỷ trọng nước tiểu',
'Hàm lượng albumin trong nước tiểu',
'Mức đường trong nước tiểu',]
num_imputer = SimpleImputer(strategy='median')

X_train[num_cols] = num_imputer.fit_transform(X_train[num_cols])
X_test[num_cols] = num_imputer.transform(X_test[num_cols])
```

Python

```
# Xử lý missing cho các cột Catgory cho X_Train và X_Test
cat_cols = ['Tình trạng hồng cầu',
'Tình trạng bạch cầu mủ',
'Tăng huyết áp',
'Tiểu đường',
'Tình trạng ăn uống']
cat_imputer = SimpleImputer(strategy='most_frequent')

X_train[cat_cols] = cat_imputer.fit_transform(X_train[cat_cols])
X_test[cat_cols] = cat_imputer.transform(X_test[cat_cols])
```

Python

Model dự đoán sớm

Vì nhóm category của chúng tôi chỉ có 2 giá trị unique nên chúng tôi có thể sử dụng labelencoder hoặc onehot

```
# Mã hóa các cột category
from sklearn.preprocessing import LabelEncoder
import joblib

encoders = {}

# Encode categorical columns
for col in cat_cols:
    le = LabelEncoder()

    x_train[col] = le.fit_transform(x_train[col])
    x_test[col] = le.transform(x_test[col])

    encoders[col] = le

# Encode target (chỉ cần fit trên y_train)
target_encoder = LabelEncoder()
y_train = target_encoder.fit_transform(y_train)
y_test = target_encoder.transform(y_test)

encoders['Phân loại'] = target_encoder

# Lưu encoder
joblib.dump(encoders, "encoders.pkl")
```

Model dự đoán sớm

Tiếp theo chúng tôi sẽ chuẩn hóa vì Logistic và KNN cần chuẩn hóa . Chúng tôi sẽ train logistic và knn với X_train đã chuẩn hóa và ngược lại với 2 mô hình cây .

```
# Chuẩn hóa dữ liệu để chuẩn bị cho Train model
from sklearn.preprocessing import MinMaxScaler

# 1. Khởi tạo scaler
scaler = MinMaxScaler()

# 2. Fit scaler trên TẬP TRAIN
scaler.fit(X_train[cols_to_scale])

# 3. Tạo bản sao dữ liệu để tránh ghi đè lên gốc
X_train_scaled = X_train.copy()
X_test_scaled = X_test.copy()

# 4. Transform các cột cần scale bằng scalar đã fit
X_train_scaled[cols_to_scale] = (variable) scaler: MinMaxScaler [scale]
X_test_scaled[cols_to_scale] = scaler.transform(X_test[cols_to_scale])
```

Model trong việc xác nhận

Đối với phần datepreprocessing chúng tôi sẽ làm giống với mô hình dự đoán sớm

```
ind_col = [col for col in df.columns if col != 'Phân loại']
dep_col = 'Phân loại'

X = df[ind_col]
y = df[dep_col]
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 0)
```

Model trong việc xác nhận

Đối với phần datepreprocessing chúng tôi sẽ làm giống với mô hình dự đoán sớm

```
# Tách cột số và cột category
num_cols = df.select_dtypes(include=['int64', 'float64']).columns
cat_cols = df.select_dtypes(include=['object', 'category']).columns

# Hàm điền giá trị Median
def impute_median(feature):
    median_value = df[feature].median()
    df[feature] = df[feature].fillna(median_value)

# Hàm điền giá trị Mode
def impute_mode(feature):
    mode_value = df[feature].mode()[0]
    df[feature] = df[feature].fillna(mode_value)

#Thực hiện imputation

# Điền cho cột số học (Median)
for col in num_cols:
    impute_median(col)

# Điền cho cột category (Mode)
for col in cat_cols:
    impute_mode(col)
```

Model trong việc xác nhận

Đối với phần datepreprocessing chúng tôi sẽ làm giống với mô hình dự đoán sớm

```
from sklearn.preprocessing import LabelEncoder
import joblib

cat_cols = [
    'Tình trạng hồng cầu', 'Tình trạng bạch cầu mủ',
    'Cụm bạch cầu mủ', 'Vi khuẩn trong nước tiểu',
    'Tăng huyết áp', 'Tiểu đường', 'Bệnh động mạch vành',
    'Tình trạng ăn uống', 'Phù chân', 'Thiếu máu'
]

target_col = "Phân loại"

encoders = {}

# Mã hóa tất cả category columns + target
for col in cat_cols + [target_col]:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    encoders[col] = le

# LƯU ENCODER
joblib.dump(encoders, "encoders.pkl")
```

Python

Model trong việc xác nhận

Đối với phần datepreprocessing chúng tôi sẽ làm giống với mô hình dự đoán sớm

```
from sklearn.preprocessing import MinMaxScaler

# 1. Khởi tạo scaler
scaler = MinMaxScaler()

# 2. Fit scaler trên TẬP TRAIN
scaler.fit(X_train[cols_to_scale])

# 3. Tạo bản sao dữ liệu để tránh ghi đè lên gốc
X_train_scaled = X_train.copy()
X_test_scaled = X_test.copy()

# 4. Transform các cột cần scale bằng scaler đã fit
X_train_scaled[cols_to_scale] = scaler.transform(X_train[cols_to_scale])
X_test_scaled[cols_to_scale] = scaler.transform(X_test[cols_to_scale])
```

Python

Train Model

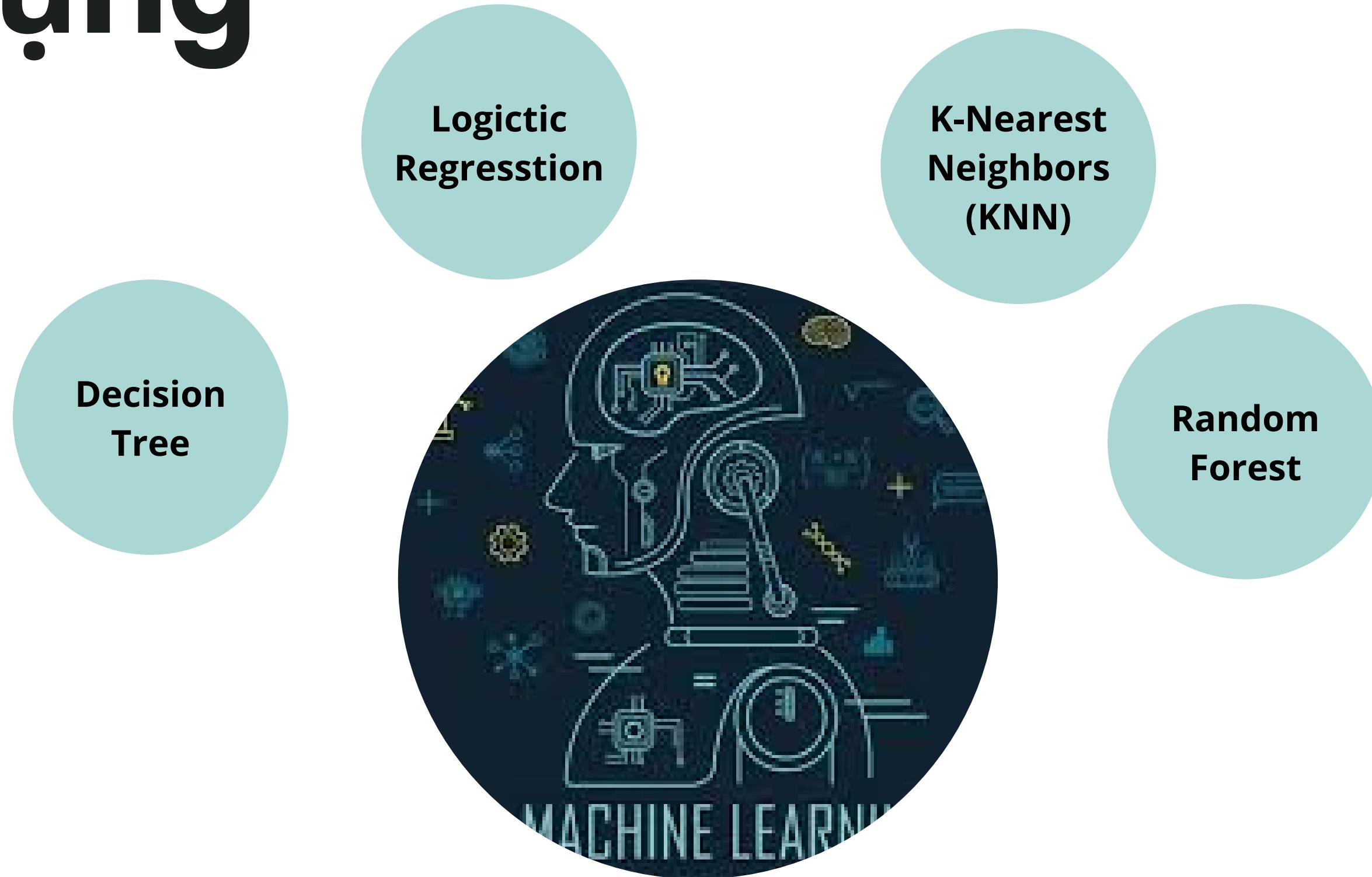
Huấn luyện mô hình (Model training) là quá trình “dạy” một mô hình học máy cách tối ưu hóa hiệu suất trên tập dữ liệu huấn luyện, bao gồm các nhiệm vụ mẫu liên quan đến các tình huống sử dụng thực tế của mô hình sau này.

Nếu dữ liệu huấn luyện gần giống với các bài toán trong thế giới thực mà mô hình sẽ phải xử lý, thì việc học được các mẫu (patterns) và mối tương quan (correlations) trong dữ liệu sẽ giúp mô hình đã được huấn luyện đưa ra dự đoán chính xác trên dữ liệu mới.

Chú ý : Bắt đầu từ bước này chúng tôi sẽ chỉ Show về mô hình dự đoán sớm , đối với mô hình xác nhận chúng tôi sẽ show code để tránh việc slide quá dài



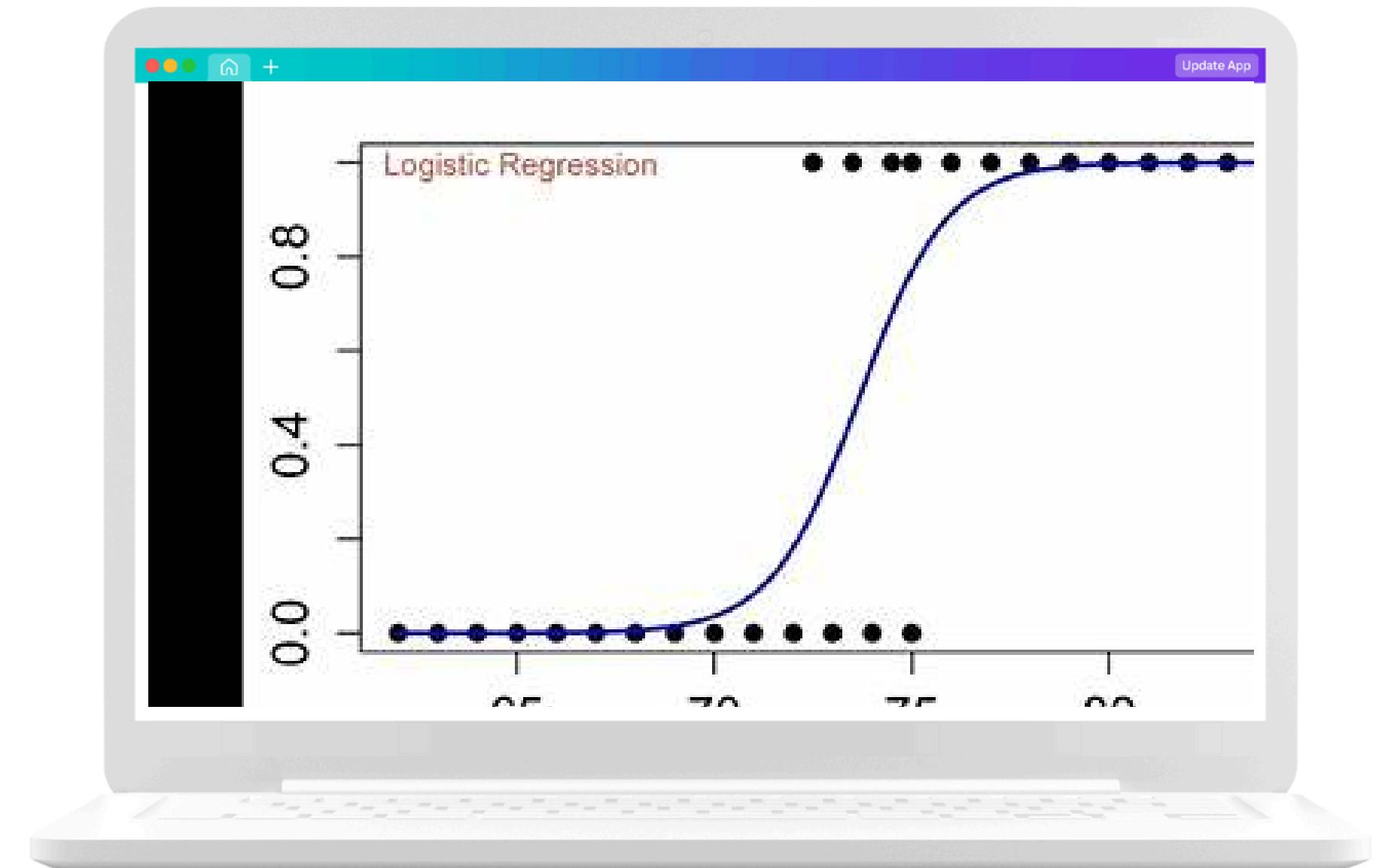
Các thuật toán sử dụng



Logistic Regression

Logistic Regression (hồi quy logistic) là một thuật toán trong machine learning dùng để giải quyết bài toán phân loại – đặc biệt là phân loại nhị phân (2 lớp), ví dụ:

- Spam / Không spam
- Bệnh / Không bệnh
- Thành công / Thất bại



Ý tưởng

- Mô hình tuyến tính sinh ra giá trị z:

$$z = w_1x_1 + w_2x_2 + \dots + b$$

- Dùng hàm Sigmoid để chuyển z thành xác suất:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Khi nào dùng?

- Khi cần bài toán phân loại nhị phân và muốn:
 - Xác suất rõ ràng
 - Mô hình nhẹ, dễ diễn giải
 - Huấn luyện nhanh

Ưu điểm

- Hiệu quả cao khi dữ liệu tuyến tính.
- Tránh overfitting tốt khi dùng regularization (L1/L2).
- Có thể giải thích trọng số → hiểu được feature nào quan trọng.

Nhược điểm

- Không xử lý được quan hệ phi tuyến phức tạp (trừ khi thêm polynomial features).

Ví dụ

Dự đoán email có phải spam (1) hay không spam (0) dựa vào:

- Tần suất xuất hiện từ "FREE"
- Tần suất xuất hiện từ "WIN"

Dữ liệu

	FREE	WIN	Spam?
	5	3	1
	0	1	0
	4	0	1
	1	2	0

Sau khi huấn luyện, mô hình Logistic Regression cho email mới:

FREE = 3, WIN = 2

kết quả:

P(spam)=0.78 → Xác suất email là spam = 78%

Nếu dùng ngưỡng 0.5 (mặc định):

Kết luận: Email là Spam

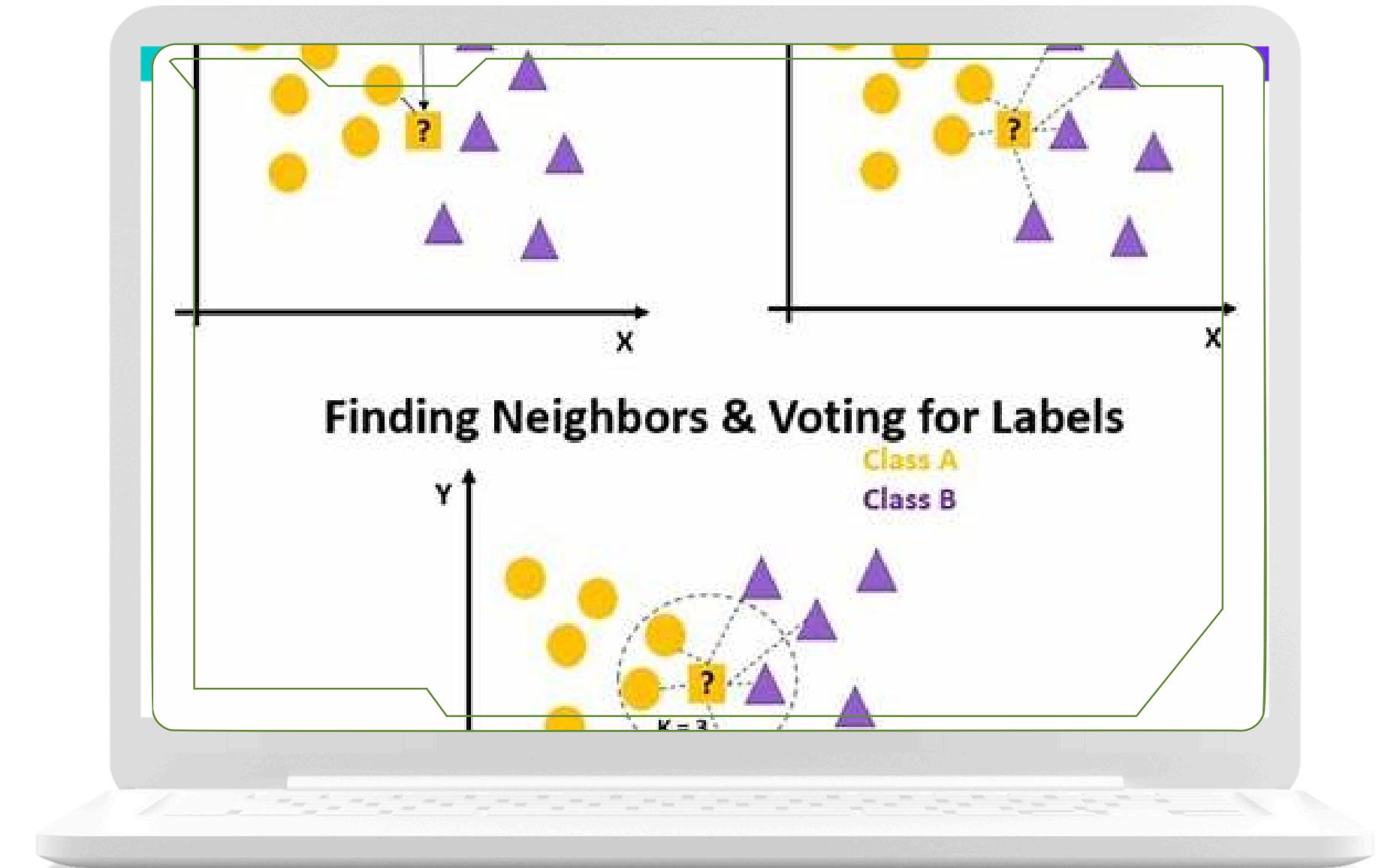
KNN (K-Nearest Neighbors)

KNN (K-Nearest Neighbors) là một thuật toán machine learning thuộc loại học có giám sát (supervised learning), dùng cho:

- Phân loại (classification)
- Hồi quy (regression) – ít phổ biến hơn

Ý tưởng chính của KNN:

Một điểm dữ liệu mới sẽ được dự đoán dựa trên K điểm gần nhất với nó trong tập dữ liệu đã biết.



Ý tưởng chính

- Thuật toán không xây dựng mô hình.
- Khi cần dự đoán dữ liệu mới, KNN sẽ:
 - a.Tính khoảng cách giữa điểm mới và tất cả điểm trong tập huấn luyện (thường dùng Euclidean).
 - b.Chọn K điểm gần nhất.
 - c.Dựa vào đa số phiếu (phân loại) hoặc trung bình (hồi quy).

Đặc điểm

- Hoạt động tốt khi dữ liệu không quá lớn và phân bố thành cụm.
- Không giả định dạng phân phối dữ liệu → phù hợp dữ liệu thực phức tạp.
- Nhưng tính toán chậm khi số mẫu lớn (vì phải đo khoảng cách với mọi mẫu).

Ví dụ

Dự đoán giới tính dựa trên chiều cao và cân nặng.

Dữ liệu

Chiều cao (cm)	Cân nặng (kg)	Giới tính
170	65	Nam
160	55	Nữ
175	70	Nam
158	50	Nữ

Cần dự đoán cho một người:

Chiều cao = 165 cm, cân nặng = 58 kg

Chọn K = 3

- Tính khoảng cách đến từng điểm

Điểm dữ liệu	Khoảng cách đến (165, 58)	Giới tính
(160, 55)	gần nhất	Nữ
(170, 65)	gần	Nam
(158, 50)	gần	Nữ
(175, 70)	xa hơn	Nam

3 điểm gần nhất \Rightarrow 2 Nữ – 1 Nam

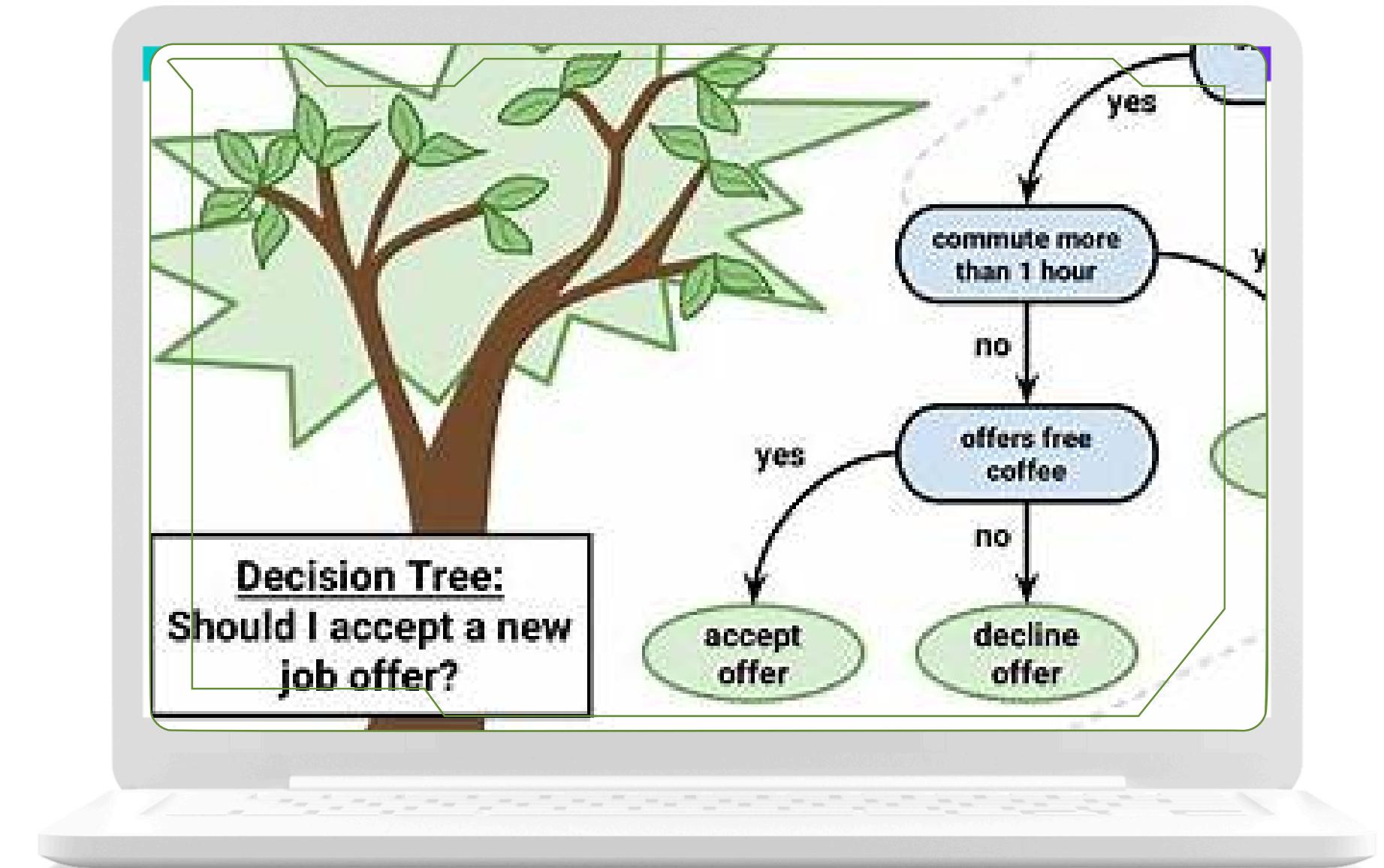
Kết luận: Người mới \rightarrow Nữ

Decision Tree

Decision Tree (Cây quyết định) là một thuật toán machine learning thuộc nhóm học có giám sát (supervised learning), dùng cho cả:

- Phân loại (classification)
- Hồi quy (regression)

Decision Tree hoạt động bằng cách:
Chia dữ liệu thành các nhánh dựa trên
các câu hỏi (điều kiện) tại từng nút để
đưa ra kết quả dự đoán.



Ví dụ

Ý tưởng

- Xây dựng một cấu trúc cây với các câu hỏi phân nhánh:
 - Ví dụ: "Thu nhập cao?" / "Tuổi < 30?"
- Mỗi nhánh giảm độ hỗn loạn của dữ liệu (dựa vào Entropy, Gini).

Ưu điểm

- Rất trực quan, dễ diễn giải (giống như hỏi–đáp).
- Không cần chuẩn hóa dữ liệu.
- Có thể xử lý dữ liệu dạng số và dạng phân loại (categorical).

Nhược điểm

- Dễ overfitting nếu cây quá sâu.
- Tính ổn định kém: dữ liệu thay đổi nhỏ → cây thay đổi nhiều.
- Quá nhạy cảm với noise

ID dự đoán khách hàng có mua sản phẩm A không:
Dữ liệu

Tuổi	Thu nhập	Đã mua?
22	Cao	Có
45	Thấp	Không
30	Trung bình	Có
40	Cao	Có

Cây có thể trông như sau:

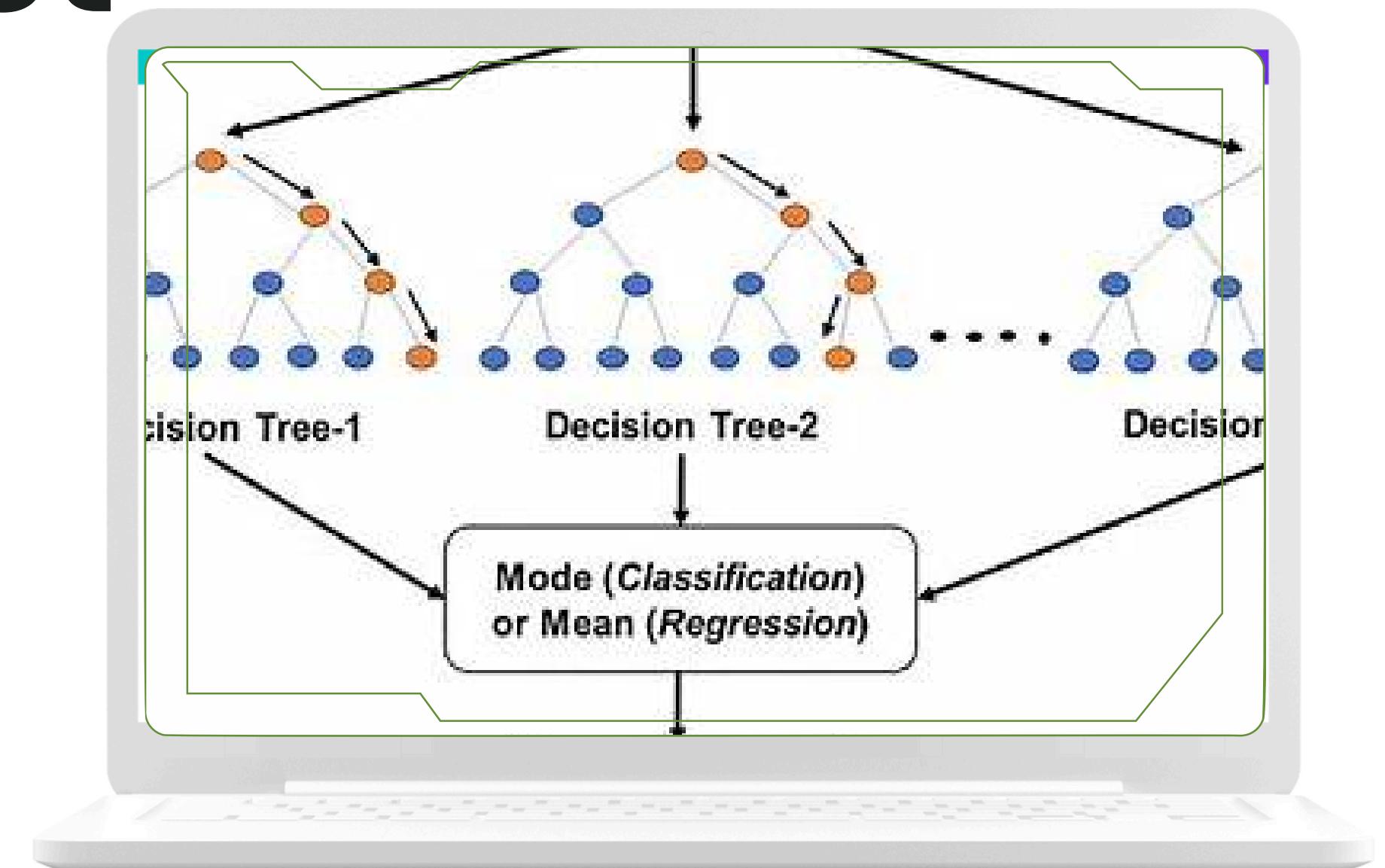
- 1.Thu nhập “Cao”? → → Dự đoán: Có mua
- 2.Nếu không: Tuổi < 35? → → Dự đoán: Có mua
- 3.Ngược lại → Không mua

Random Forest

Random Forest là một thuật toán ensemble learning gồm nhiều cây quyết định (Decision Trees) được xây dựng ngẫu nhiên và sau đó bỏ phiếu vote nếu là bài toán phân loại và trung bình nếu là bài toán hồi quy để đưa ra dự đoán cuối cùng.

Nói đơn giản:

Random Forest = nhiều Decision Tree + vote để ra kết quả chính xác hơn.



Ý tưởng

- Là phương pháp Ensemble Learning dựa trên Bagging:
 - Tạo nhiều Decision Tree (thường hàng chục đến hàng trăm cây).
 - Mỗi cây được huấn luyện bằng:
 - Một phần mẫu dữ liệu (lấy ngẫu nhiên – Bootstrap).
 - Một phần đặc trưng ngẫu nhiên.
- Khi dự đoán:
 - Với bài toán phân loại → bỏ phiếu số đông
 - Với bài toán hồi quy → lấy trung bình

Ưu điểm

- Giảm đáng kể overfitting so với một cây đơn.
- Giúp giảm mạnh variance và triệt tiêu rất tốt với noise
- Mạnh trong hầu hết tình huống thực tế.
- Hoạt động tốt trên dữ liệu lớn với nhiều đặc trưng.

Nhược điểm

- Ít trực quan (không thể xem 100 cây).
- Tốn tài nguyên tính toán.

Ví dụ

Dự đoán khách hàng có trả nợ hay không (1=có, 0=không).

Giả sử Random Forest gồm 5 cây (mini ví dụ).

Dữ liệu đầu vào cho khách mới

- Thu nhập: Trung bình
- Tuổi: 30
- Điểm tín dụng 680

5 cây dự đoán như sau

Cây số	Dự đoán
1	1 (Có trả)
2	1
3	0
4	1
5	1

→ 4 cây chọn 1, chỉ 1 cây chọn 0.

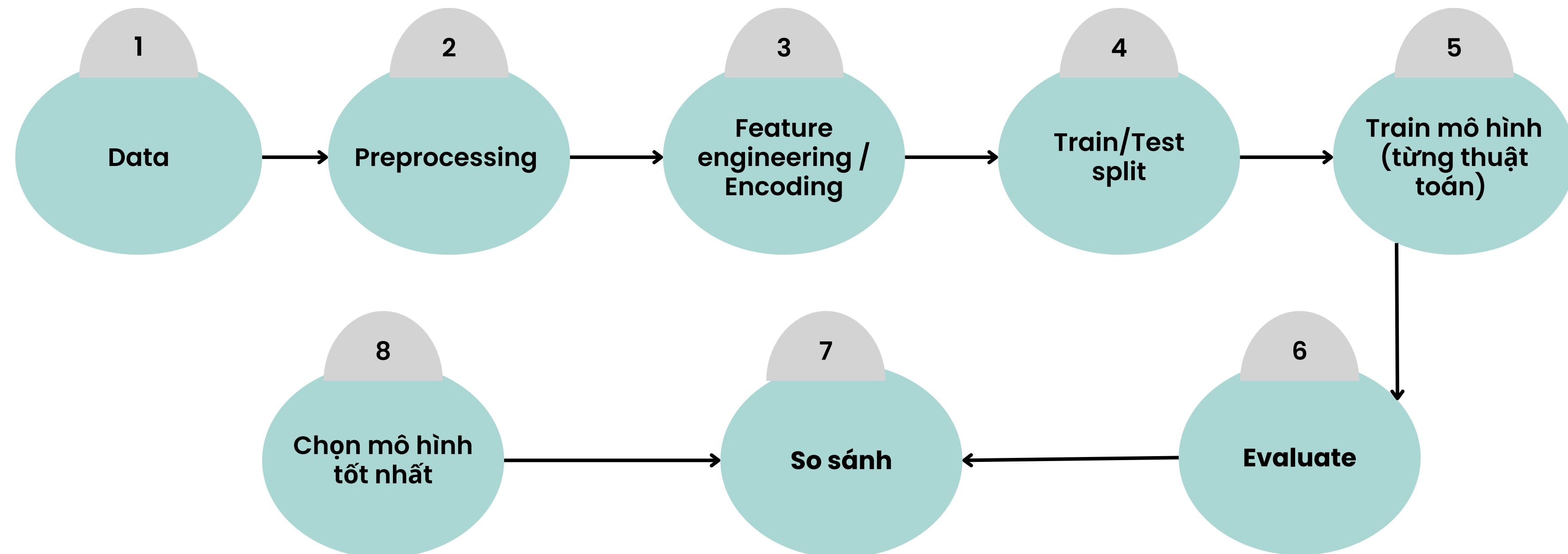
Random Forest kết luận: 1 (Có trả nợ)

Xác suất $\approx 4/5 = 80\%$

So sánh tổng quan

Thuật toán	Loại bài toán	Ưu điểm	Nhược điểm	Ví dụ ứng dụng
KNN	Phân loại / hồi quy	Dễ hiểu, không cần mô hình	Chậm với dữ liệu lớn	Gợi ý phim giống nhau
Decision Tree	Phân loại / hồi quy	Dễ diễn giải	Dễ overfit	Phân loại rủi ro khách hàng
Random Forest	Ensemble	Mạnh, ổn định, chống overfitting	Ít trực quan	Chấm điểm rủi ro tín dụng
Logistic Regression	Phân loại nhị phân	Cho ra xác suất, nhanh	Khó với dữ liệu phi tuyến	Phát hiện spam, dự đoán bệnh

Quy trình xây dựng mô hình



Model dự đoán sớm

Chúng tôi đã thử với logistic mặc định sklearn . Sau đó chúng tôi đã chỉ định các hyperparameter cho đầu vào và đây là kết quả

```
# Logistic
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

log_clf = LogisticRegression(
    penalty='l2',
    C=1.0,
    solver='liblinear',
    max_iter=200
)

# Train mode (variable) x_train_scaled: Any
log_clf.fit(x_train_scaled, y_train)

log_acc = accuracy_score(y_test, log_clf.predict(x_test_scaled))

print(f"Training Accuracy of Logistic Regression is {accuracy_score(y_train, log_clf.predict(x_train_scaled))}")
print(f"Test Accuracy of Logistic Regression is {log_acc} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, log_clf.predict(x_test_scaled))}\n")
print(f"Classification Report :- \n {classification_report(y_test, log_clf.predict(x_test_scaled))}")
```

Model dự đoán sớm

Chúng tôi đã thử với logistic mặc định sklearn . Sau đó chúng tôi đã chỉ định các hyperparameter cho đầu vào và đây là kết quả

Training Accuracy of Logistic Regression is 0.971875

Test Accuracy of Logistic Regression is 0.975

Confusion Matrix :-

```
[[48  2]
 [ 0 30]]
```

classification Report :-

	precision	recall	f1-score	support
0	1.00	0.96	0.98	50
1	0.94	1.00	0.97	30
accuracy			0.97	80
macro avg	0.97	0.98	0.97	80
weighted avg	0.98	0.97	0.98	80

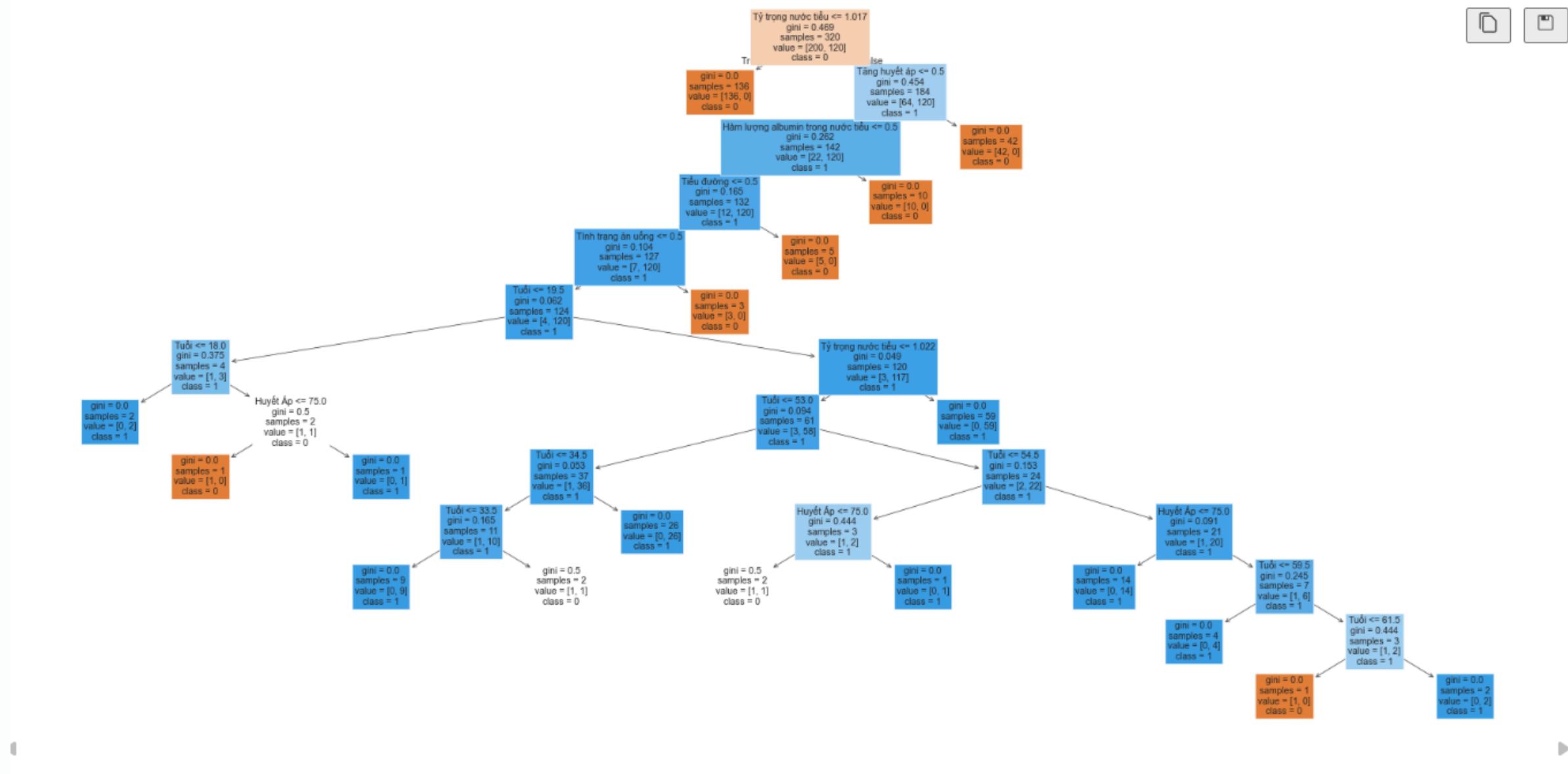
Model dự đoán sớm

Đối với các model khác chúng tôi làm tương tự và đây là kết quả của 4 mô hình trên 2 tiêu chí đánh giá Accuracy và Confusion matrix . Nhưng chúng tôi sẽ không chỉ căn cứ vào đây để chọn mô hình tốt nhất bởi vì nó có thể có tính may rủi .

Model	Accuracy train	Accuracy test	Precision	Recall	F1-Score	AUC
Logistic	0.97188	0.975	0.9375	1	0.96774	0.98133
Dtree	0.99375	0.9875	0.96774	1	0.98361	0.99
Randomf	0.9875	0.975	0.9375	1	0.96774	0.99867
Knn	0.9875	0.975	0.9375	1	0.96774	0.99867

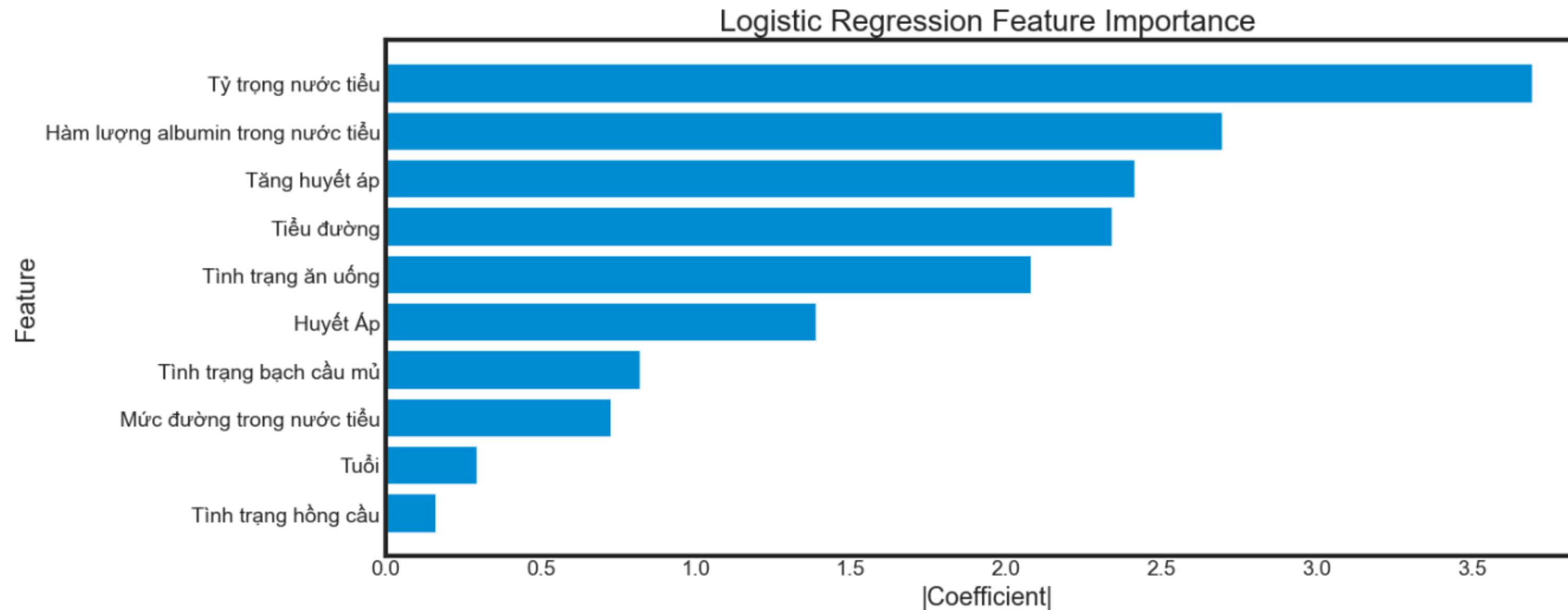
Model dự đoán sóm

Feature important và sơ đồ cây và Roc&AUC



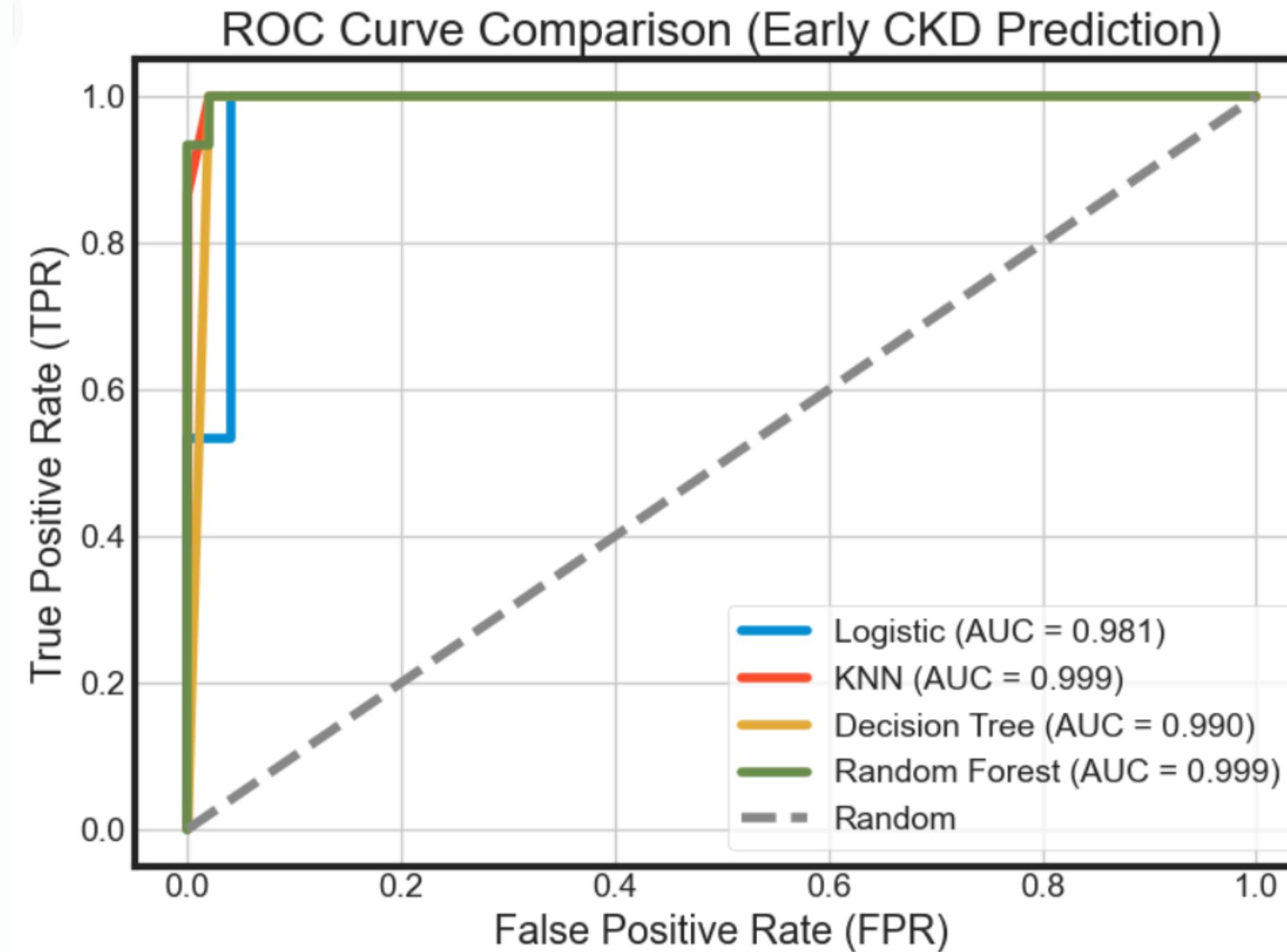
Model dự đoán sớm

Feature important và sơ đồ cây và Roc&AUC



Model dự đoán sớm

Feature important và sơ đồ cây và Roc&AUC



Đề Xuất K-Fold

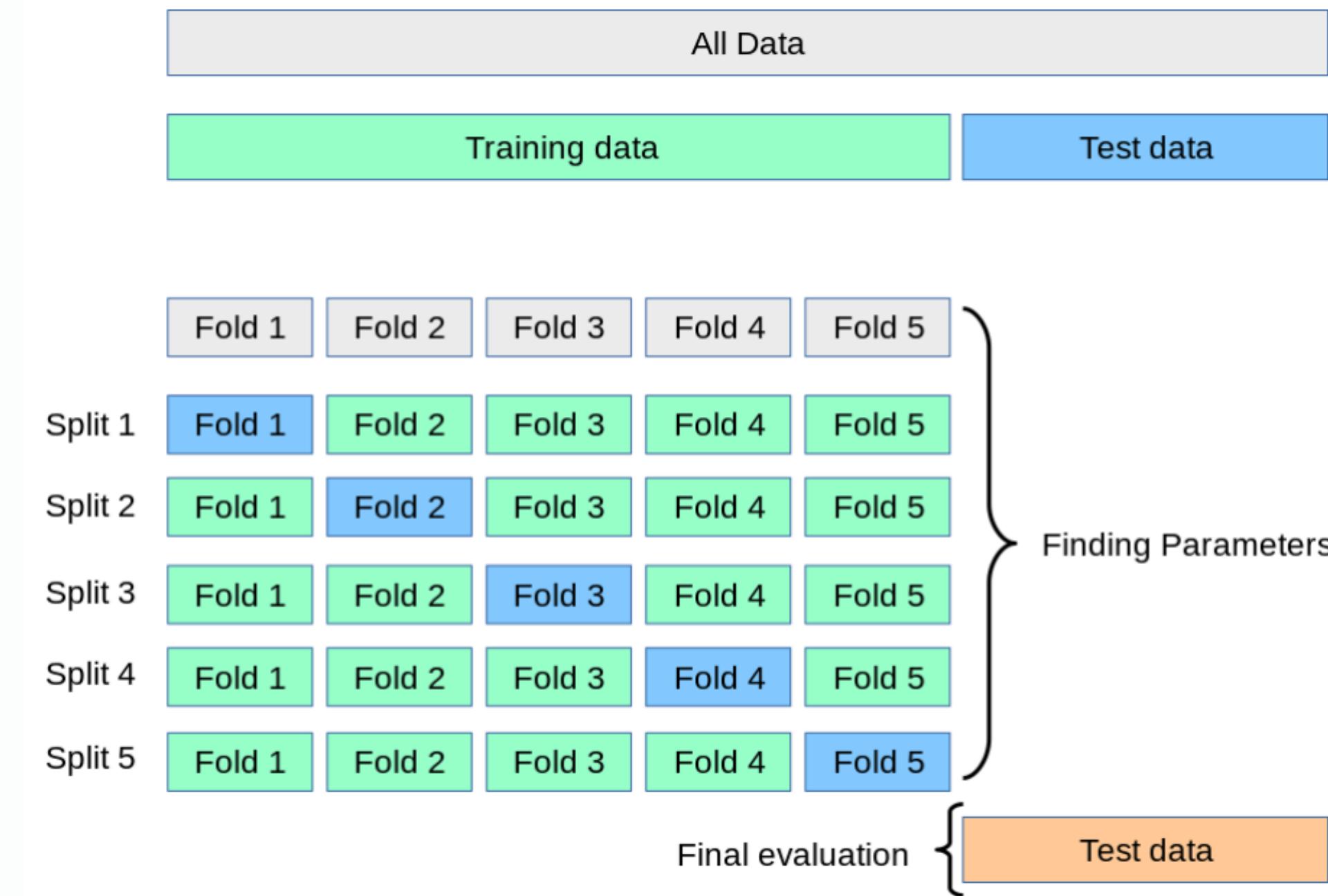
Đối với data nhỏ như của chúng tôi . Không cho phép việc chia Train validation test được .

Là vì có thể một số điểm dữ liệu có ích cho quá trình train đã bị bạn ném vào để làm validation, test và model không có cơ hội học điểm dữ liệu đó. thậm chí, đôi khi do ít dữ liệu nên có một vài class chỉ có trong validation, test mà không có trong train (do việc chia train, val là hoàn toàn ngẫu nhiên) dẫn đến một kết quả tồi tệ khi validation và test.

K-Fold CV sẽ giúp chúng ta đánh giá một model đầy đủ và chính xác hơn khi chúng ta có một tập dữ liệu không lớn. Để sau đó chúng ta đưa ra quyết định model đó có phù hợp với dữ liệu, bài toán hiện tại hay không để mà đưa ra next action.

Đề Xuất K-Fold

Minh họa K-fold



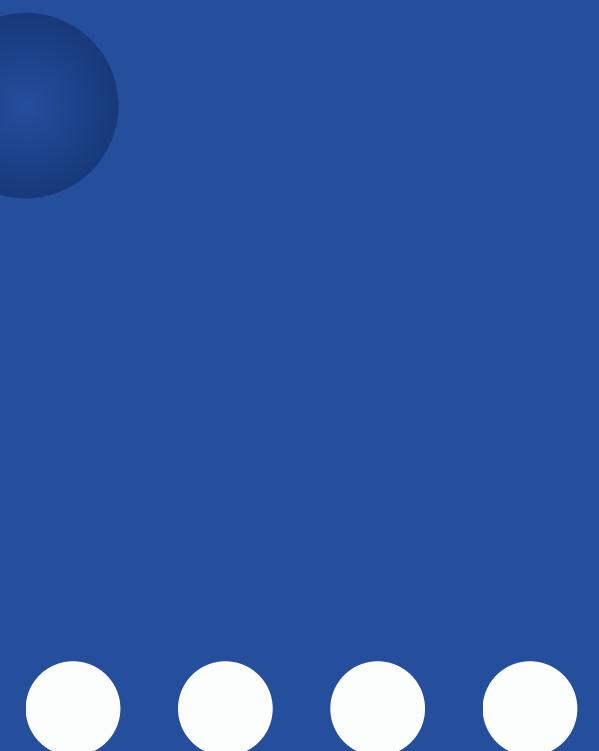
Model dự đoán sớm

Chúng tôi train với K= 5 trên 4 mô hình và đây là kết quả trung bình . Qua đây ta có thể nhận xét rằng logistic là mô hình tốt nhất với dữ liệu của chúng tôi .

Mô hình	Mean Train Accuracy	Mean Validation Accuracy	Mean AUC	Std AUC
Logistic Regression	0.9852	0.9750	0.9946	0.0036
KNN	0.9969	0.9469	0.9773	0.0135
Decision Tree	0.9313	0.9313	0.9442	0.0253
Random Forest	0.9766	0.9719	0.9951	0.0070



Đánh giá sơ bộ



Đánh giá

Thông qua việc sử dụng K-fold và kết quả của các mô hình : Chúng tôi nhận thấy rằng kết quả rất tốt nhỉn sơ qua thì không thấy dấu hiệu của overfit hay các vấn đề khác

Có thể hiểu kết quả đó có thể do dataset của chúng tôi và căn bệnh này nó có tính tách biệt khá rõ ràng nên mô hình học quá dễ có thể hiểu vì Auc cho thấy đang rất cao

Nhưng để chắc chắn về kết quả hơn chúng tôi đề xuất sử dụng hai phương pháp là Shuffled labels và biểu đồ Learning curve



Chứng minh kết quả



www.reallygreatsite.com

Shuffled label

Shuffled labels (xáo trộn nhãn) là kỹ thuật:

Giữ nguyên dữ liệu đầu vào X, nhưng xáo trộn ngẫu nhiên nhãn y
hoặc có thể hiểu là cắt đứt hoàn toàn mối quan hệ giữa dữ liệu và nhãn

Mục đích để xem mô hình có đang học thật hay không hay chỉ học quy luật giả

Ban đầu:

SCSS

x (đặc trưng) → y (CKD / not CKD)

Sau khi shuffle:

CSS

x (giữ nguyên) → y (bị trộn ngẫu nhiên)

Shuffled label

Ta có thể thấy sau khi hủy đi mối liên kết giữa x và y mô hình không học được gì
Dự đoán trên test là Random

```
accs.append(accuracy_score(y_test, y_test_pred))

print("Mean shuffled AUC:", np.mean(aucs))
print("Std shuffled AUC:", np.std(aucs))
print("Mean shuffled Accuracy:", np.mean(accs))
```

Mean shuffled AUC: 0.4759866666666667

Std shuffled AUC: 0.2143666014616601

Mean shuffled Accuracy: 0.5825

Learning curve

Learning curve là sự thay đổi của accuracy (hoặc loss) theo số lượng dữ liệu huấn luyện

- ◆ Learning curve giúp ta biết điều gì?

Vấn đề

Overfitting

Underfitting

Thiếu dữ liệu

Mô hình tốt

Nhìn vào learning curve

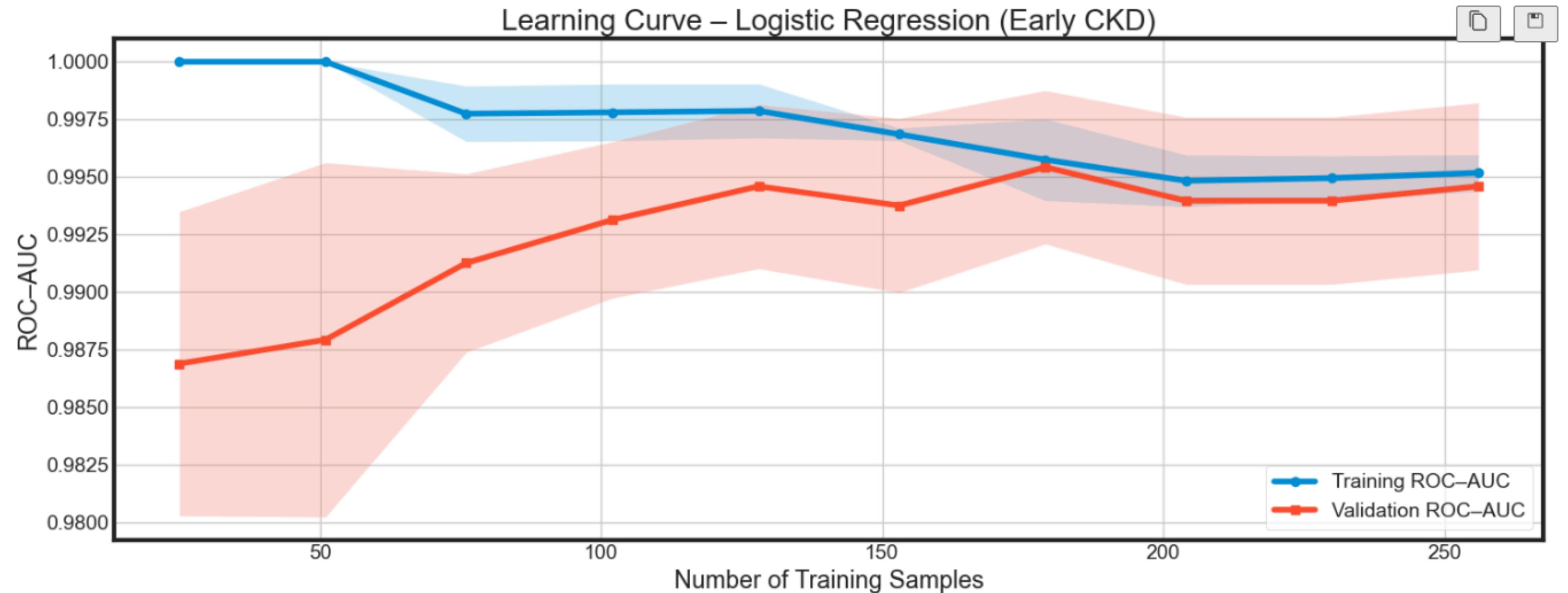
Train cao – Test thấp

Cả train & test đều thấp

Test tăng dần khi thêm data

Train & test cao và gần nhau

Learning curve



Kết luận



Độ tin cậy của mô hình

```
    print(f"Độ tin cậy: {prob_not_ckd*100:.2f}%")  
  
print("\nChi tiết xác suất")  
print(f"P(CKD): {prob_ckd:.4f}")  
print(f"P(NOT CKD): {prob_not_ckd:.4f}")
```

✓ 0.0s

KẾT QUẢ DỰ ĐOÁN

Bệnh nhân không mắc CKD

Độ tin cậy: 93.47%

Chi tiết xác suất

P(CKD): 0.0653

P(NOT CKD): 0.9347

Tài liệu tham khảo

1. Machinelearningcoban.com
2. Sách bài tập lớp AIO
3. Wallstreetmojo.com
4. arxiv.org
5. studocu.vn

Demo mô hình dự đoán sớm

 Dự đoán sớm bệnh suy thận mạn tính (CKD)

Tuổi	Huyết áp (mmHg)
<input type="text"/>	<input type="text"/>
Tỷ trọng nước tiểu	Albumin trong nước tiểu
<input type="text"/>	<input type="text"/>
Đường trong nước tiểu	Tình trạng hồng cầu
<input type="text"/>	<input type="text"/> Bình thường
Tình trạng bạch cầu mù	Tăng huyết áp
<input type="text"/> Bình thường	<input type="text"/> Không
Tiểu đường	Tình trạng ăn uống
<input type="text"/> Không	<input type="text"/> Tốt

 [Dự đoán nguy cơ suy thận](#)

 KẾT LUẬN: **KHÔNG CÓ DẤU HIỆU SUY THẬN**
Độ tin cậy: **88.47%**

Demo mô hình dự đoán sớm

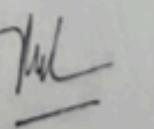

BỆNH VIỆN CHỢ RẪY
 HUYẾT HỌC - SINH HÓA - TTTM - VI SINH
 Địa chỉ: 201B Nguyễn Chí Thanh, Q5, TP.HCM
 Điện thoại: (08) 3842 0000


 VIỆT NAM NATIONAL UNIVERSITY
 HO CHI MINH CITY

Họ tên: LÊ ĐỨC MẠNH	Năm sinh: 2002	Giới tính: Nam
Mã ID: 202305260001	Loại mẫu: Máu Nước tiểu	
Đài tượng: Thu phí	Chất lượng mẫu:	
BS chỉ định: Lư Thị Mỹ Dung	T/G lấy mẫu:	26/05/23 07:24
Noi gửi: PK.Nội Thận	T/G nhận mẫu:	26/05/23 07:50
Chẩn đoán: Suy thận mạn, giai đoạn 5	T/G in kết quả:	26/05/23 10:21

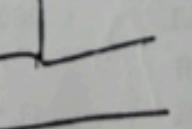
Tên xét nghiệm	Kết quả	Đơn vị	Chi số bình thường	Thiết bị - Mã QT
<i>Sau A.Mang</i> , 500 5 thùng Vỏ dưới	B.U.N 67 Creatinin 7.34 eGFR (CKD-EPI) 9.62	mg/dL mg/dL mL/min/1. 73m ²	(7 - 20) (0.7 - 1.5) ≥90	
* Ion đồ máu				
Na+	138	mmol/L	(135 - 150)	
K+	5.3	mmol/L	(3.5 - 5.5)	
Cl-	105	mmol/L	(98 - 107)	
Ca TP	2.2	mmol/L	(2.2 - 2.6)	
* Phân tích NT (10 ts)				
pH	6.5		(5.0 - 8.0)	
S.G	1.010		(1.003 - 1.030)	
Glucose	neg	mg/dL	Âm tính	
Protein NT	+++ 300	mg/dL	Âm tính / vết	
Bilirubin	neg	mg/dL	Âm tính	
Urobilinogen	norm	mg/dL	(0.1 - 1.0)	
Ketone	neg		Âm tính	
Blood	25	RBC/uL	Âm tính	
Leukocytes	neg	WBC/uL	Âm tính	
Nitrite	neg		Âm tính	

Khoa Huyết Học



CN. Nguyễn Thị Thảo

Khoa Sinh Hóa



KTY.III Nguyễn Thị Dung

Người lấy mẫu: Lê Thị Thu Xương(RTQ) 2/2

Review code

Thank
You