ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

MÔN HỌC: LẬP TRÌNH PYTHON CHO MÁY HỌC

ĐỀ TÀI: DECISION TREE CLASSIFIER

Giảng viên hướng dẫn	**:**	**TS.Nguyễn Vinh Tiệp**
Sinh viên thực hiện	**:**	**Đỗ Trọng Khánh – 19521676**
		Trịnh Công Danh - 19521326
		Võ Phạm Duy Đức – 19521383
Lớp	**:**	**CS116.M12.KHCL**

Thành phố Hồ Chí Minh, ngày 25 tháng 12 năm 2021

MÔ TẢ ĐỒ ÁN

Đồ án cuối kì của nhóm em lần này đề tài là về **Decision Tree Classifier**. Nhóm sẽ tiếp cận mô hình theo hướng **gray box**, tìm hiểu các hoạt động của mô hình, cách xây một cây quyết định như thế nào và ý nghĩa của các tham số trong mô hình. Sau đó áp dụng những kiến thức tìm hiểu được vào dữ liệu của nhóm.

Chúng em xin gửi lời cảm ơn chân thành đến thầy **Nguyễn Vinh Tiệp** đã quan tâm, hướng dẫn, truyền đạt những kiến thức và kinh nghiệm cho chúng em trong suốt thời gian học tập môn **Lập trình Python cho Máy Học**.

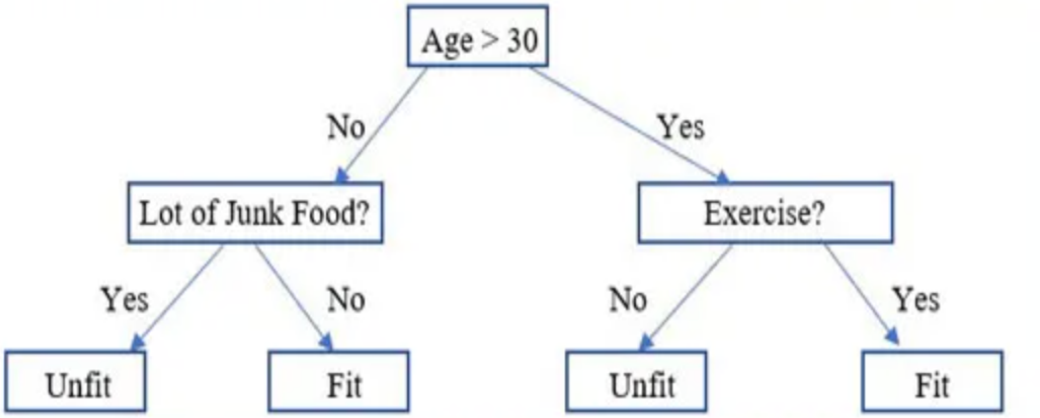
Trong quá trình làm đồ án môn không tránh khỏi được những sai sót, chúng em mong nhận được sự góp ý của thầy và các bạn để được hoàn thiện hơn.

MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU VỀ MÔ HÌNH

1. Giới thiệu về thuật toán Decision Tree

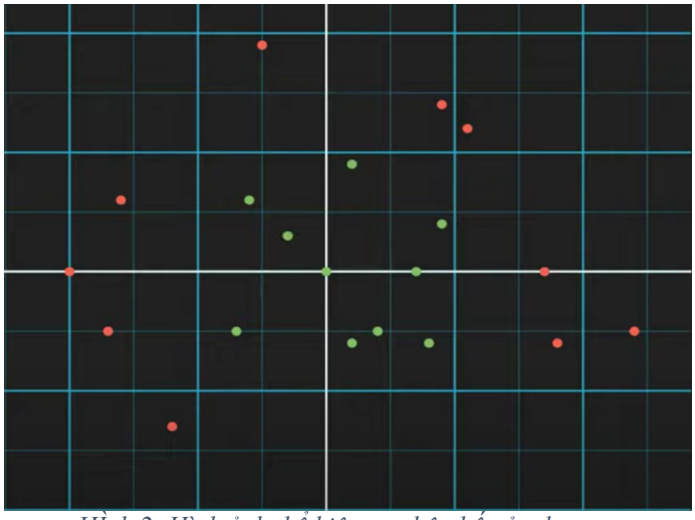
- Thuật toán **Decision Tree** là thuật toán **supervised learning** (học có giám sát), nó có thể giải quyết được bài toán **regression** (hồi quy) và **classification** (phân loại).
- **Decision Tree** nói chung là **một cây nhị phân** chia dữ liệu một cách đệ quy cho đến khi chúng ta còn lại các nút lá thuần tuý và nốt lá đó đại diện cho một lớp chúng ta cần phân loại.



HÌnh 1: Hình ảnh minh hoạ Decision Tree (Cây quyết định).

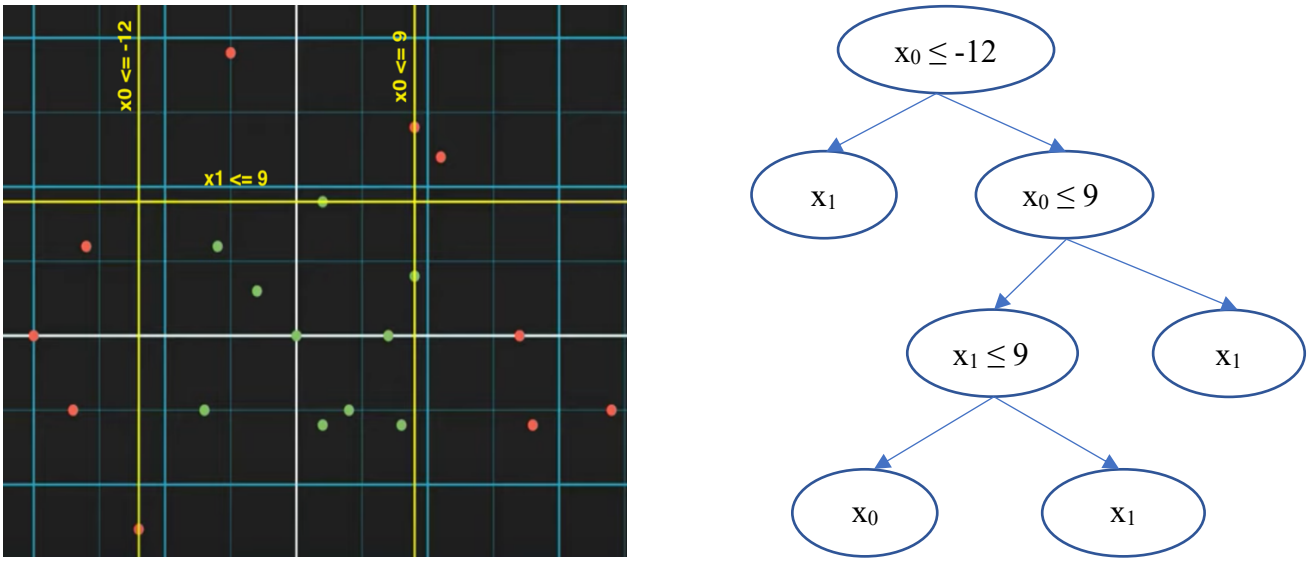
2. Cách xây dựng Decision Tree Classifier (Cây quyết định)

- Đầu tiên ta có một bộ dữ liệu gồm hai lớp: x_0 (màu xanh), x_1(màu đỏ)



HÌnh 2: Hình ảnh thể hiện sự phân bố của dataset.

- Như ***Hình 2*** chúng ta thấy rằng các lớp được phân bố không tuyến tính vì thế ta thể vẽ một đường để phân tách 2 dữ liệu đó.
- Về cơ bản Decision Tree có 2 phần chính: decision node và leaf node, ở mỗi node sẽ chứa những điều kiện giúp ta chia nhỏ những node đó sau đó dựa vào vào những node sau cùng đó ta để dự đoán những những điểm dữ liệu mới.



HÌnh 3: Hình ảnh thể hiện sự phân chia của cây quyết định.

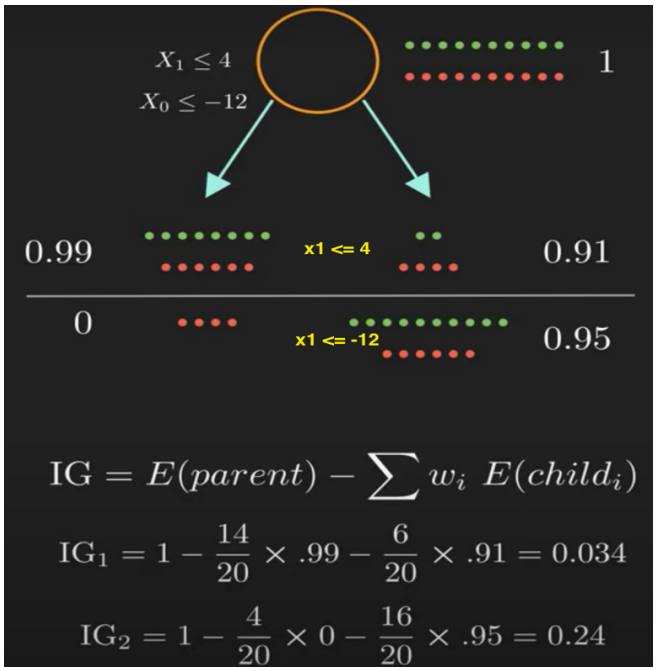
- Đầu tiên với decision node với điều kiện $x_o \leq -12$ ta sẽ có được đường thẳng đại diện như hình trên.

  + Ta có thể thấy được những điểm nằm bên trái đường thẳng (thoả điều kiện) chứa toàn điểm x_1 do đó leaf node này đã thuần tuý nên không cần chia nhỏ nữa

  + Tiếp theo là những điểm nằm bên phải đường thẳng (không thoả điều kiện) chứa cả 2 điểm x_0 và x_1 nên ta phải tiếp tục chia nhỏ với điều kiện ($x_0 \leq 9$).

  + Làm tương tự với điều kiện $x_0 \leq 9$ cho đến khi nào ta nhận được các leaf node thuần tuý như cây nhị phân được vẽ ở trên.

- Vậy làm thế nào để chọn được điều kiện tối ưu cho decision node?



HÌnh 4: Hình ảnh mô tả cách tính Infomation Gain.

- Với mỗi **leaf node** ta sẽ tính được **Entropy** cho mỗi node với công thức:

$$\textbf{Entropy} = \Sigma - p_i \log(p_i)$$

p_i = probability of class i

- **Entropy** càng cao thì số lần chia các **leaf node** sẽ càng nhiều.
- Tiếp theo để tính được số **information gain** tương ứng với mỗi sự phân chia ta phải trừ các **Entropy(child)** từ **Entropy(parent)** như hình bên.
- Sau đó ta sẽ do sánh 2 **information gain** và chọn IG lớn hơn: $IG_1 < IG_2 \Rightarrow x_0 <= -12$.

CHƯƠNG 2: ƯU VÀ NHƯỢC ĐIỂM CỦA MÔ HÌNH

1. Ưu điểm

- Mô hình sinh ra các quy tắc dễ hiểu cho người đọc, tạo ra bộ luật với mỗi nhánh lá là một luật của cây.
- Dữ liệu đầu vào có thể là là dữ liệu missing, không cần chuẩn hóa hoặc tạo biến giả.
- Có thể làm việc với cả dữ liệu số và dữ liệu phân loại.
- Có thể xác thực mô hình bằng cách sử dụng các kiểm tra thống kê.
- Có khả năng là việc với dữ liệu lớn.

2. Nhược điểm

- Mô hình cây quyết định phụ thuộc rất lớn vào dữ liệu của bạn. Thậm chí, với một sự thay đổi nhỏ trong bộ dữ liệu, cấu trúc mô hình cây quyết định có thể thay đổi hoàn toàn.
- Cây quyết định hay gặp vấn đề overfitting.

CHƯƠNG 3: THAM SỐ CỦA MÔ HÌNH

1. Tham số của mô hình Decision Tree

DecisionTreeRegressor(*, criterion='gini', max_depth=None,
min_samples_split=2, max_leaf_nodes=None)

- Trong đó:

- **criterion**: Là hàm số để đo lường chất lượng phân chia ở mỗi node. Có hai lựa chọn là *gini* và *entropy*.
- **max_depth**: Độ sâu tối đa cho một cây quyết định. Đối với mô hình bị overfitting thì chúng ta cần gia tăng độ sâu và vị khớp thì giảm độ sâu.
- **min_samples_split**: Kích thước mẫu tối thiểu được yêu cầu để tiếp tục phân chia đối với node quyết định. Được sử dụng để tránh kích thước của node lá quá nhỏ nhằm giảm thiểu hiện tượng overfitting.
- **max_leaf_nodes**: Số lượng các node lá tối đa của cây quyết định. Thường được thiết lập khi muốn kiểm soát hiện tượng overfitting.

2. Tunning siêu tham số cho mô hình Decision Tree

- Sử dụng phương pháp **Grid Search** để tìm **Hyperparameter** (siêu tham số) cho mô hình.

```
from sklearn.pipeline import Pipeline
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

parameters = {
    'clf__max_depth':[2, 3, 4],
    'clf__criterion':['gini', 'entropy'],
    'clf__min_samples_split': [2, 4],
    'clf__min_samples_leaf': [5, 10, 13],
    'clf__max_leaf_nodes': [8, 16, 32],
}


pipeline = Pipeline(
    steps=[("clf", DecisionTreeClassifier())]
)

gscv = GridSearchCV(pipeline, parameters, cv=5, n_jobs=12, scoring='accuracy',
                    return_train_score=True, error_score=0, verbose=3)
gscv.fit(X, y)
```

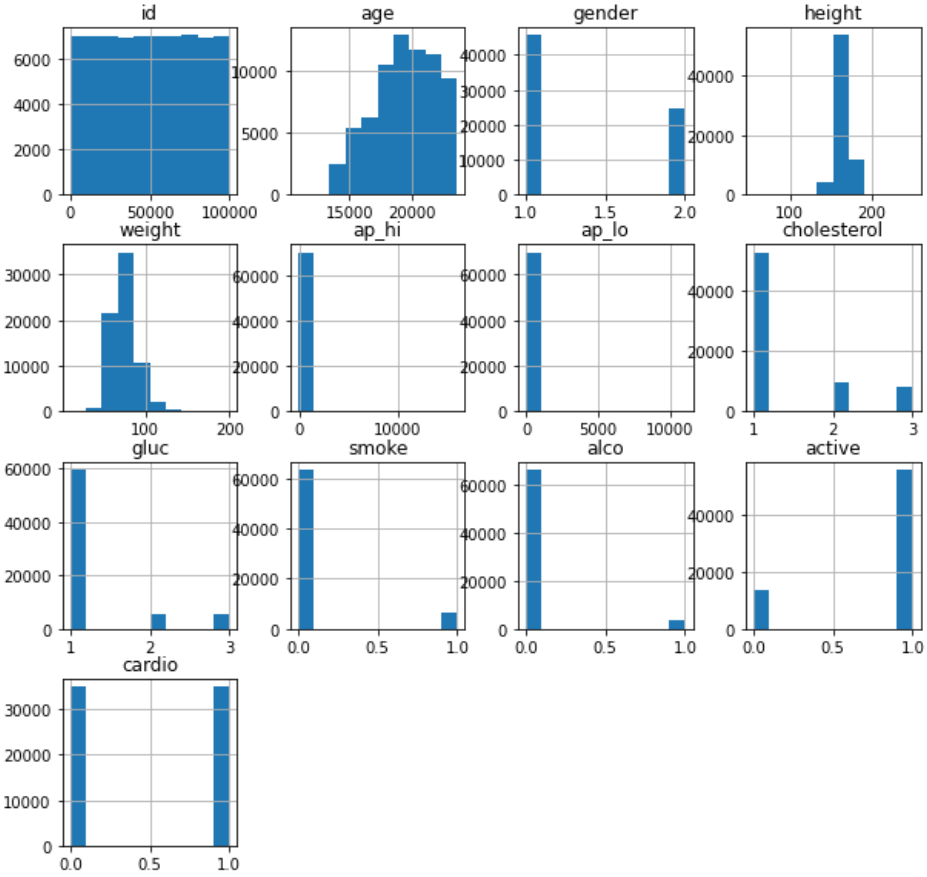
HÌnh 5: Hình ảnh mô tả phương pháp Grid Search dùng dề Tunning siêu tham số.

CHƯƠNG 4: ÁP DỤNG MÔ HÌNH VÀO DỮ LIỆU CỦA NHÓM

1. Cardiovascular Disease Predict

a. Mô tả bộ dữ liệu

- Thông tin bộ dữ liệu: Cardiovascular Disease Dataset.
- Jupyter Notebook của nhóm: Cardiovascular_Disease_Predict.ipynb
- Github của nhóm: CS116.M12.KHCL



HÌnh 6: Mô tả số liệu của các cột trong dữ liệu

- Bộ dữ liệu gồm có 12 trường dữ liệu (trừ cột id) được trực quan hoá ở *Hình 6:*
  - age: tuổi được tính theo ngày.
  - gender: giới tính (1: women, 2 men).
  - height: chiều cao.
  - weight: cân nặng.
  - ap_hi: chỉ số huyết áp thấp nhất trong mạch máu.

- ap_io: chỉ số huyết áp tối đa trong mạch máu.
- cholesterol: lượng cholesterol trong máu (1: normal, 2: above normal, 3: well above normal).
- gluc: lượng gluc trong máu (1: normal, 2: above normal, 3: well above normal).
- smoke: bệnh nhân có hút thuốc hay không.
- alco: bệnh nhân có uống rượu hay không.
- active: lối sống tích cực hay tiêu cực.
- cardio: bệnh nhân có mắc bệnh về tim mạch hay không.

b. Xử lý dữ liệu

- Bước 1: Xoá cột id và các hàng có chỉ số giống nhau.

```
df = pd.DataFrame(data)
df = df.drop(['id'], axis=1)
```

- Bước 2: Thêm cột BMI (chỉ số khối của cơ thể) vào dataset

```
df["bmi"] = (df["weight"] / (df["height"] /100)**2).round(1)
df.drop(["weight","height"],axis = 1,inplace = True)

df = df[(df["bmi"]>10) & (df["bmi"]<100)]
```

- Bước 3: Sử dụng **Label Binarizer** để chyển dữ liệu của cột gender, cholesterol và gluc về nhị phân.

```
df['gender'] =  df['gender'] % 2

lb = preprocessing.LabelBinarizer()
df_cholesterol = df['cholesterol']
df_gluc = df['gluc']
df_cholesterol = lb.fit_transform(df_cholesterol)
df_gluc = lb.fit_transform(df_cholesterol)

df = df.drop(["cholesterol", "gluc"], axis=1)
```

- Bước 4: Chuyển dữ liệu của cột age về năm và chia X_data và y_data.

```
df['age'] =  df['age'] // 365
y_data = df['cardio']
X_data = df.drop(['cardio'], axis=1)
```

c. Huấn luyện và đánh giá mô hình

- Không sử dụng KFlod: 65 %

```
X_train, X_test, y_train, y_test = train_test_split(X_data,
y_data, test_size=0.33, random_state=42)

clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print(classification_report(y_pred, y_test))
```

	precision	recall	f1-score	support
0	0.69	0.63	0.66	12539
1	0.60	0.66	0.63	10542
accuracy			0.65	23081
macro avg	0.65	0.65	0.64	23081
weighted avg	0.65	0.65	0.65	23081

- Sử dụng phương pháp **KFold**: 63.95 %

```
kfold = KFold(n_splits=5, shuffle=True, random_state=1)
accuracy_list = []

for train, test in kfold.split(X_data, y_data):
    clf = DecisionTreeClassifier()
    X_train, X_test = X_data[train], X_data[test];
    y_train, y_test = y_data[train], y_data[test]

    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy_list.append(accuracy_score(y_test, y_pred))

print("Accuracy: ", np.mean(accuracy_list) * 100)
```

d. Điều chỉnh siêu tham số cho mô hình

```
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
```

```
# Chia tập train và test
X_train, X_test, y_train, y_test = train_test_split(X_data,
y_data, test_size=0.33, random_state=42)

parameters = {
    'clf__max_depth': [2, 3, 4],
    'clf__criterion': ['gini', 'entropy'],
    'clf__min_samples_split': [2, 4],
    'clf__max_leaf_nodes': [8, 16, 32],
}

pipeline = Pipeline(steps=[("clf", DecisionTreeClassifier())])

tree_grid = GridSearchCV(pipeline, parameters, cv=5, n_jobs=12,
scoring='accuracy')
tree_grid.fit(X_train, y_train)
```

- Độ chính xác đạt được: **73%** tăng được **10%** so với chưa điều chỉnh tham số.

	precision	recall	f1-score	support
0	0.74	0.73	0.73	11821
1	0.72	0.74	0.73	11260
accuracy			0.73	23081
macro avg	0.73	0.73	0.73	23081
weighted avg	0.73	0.73	0.73	23081

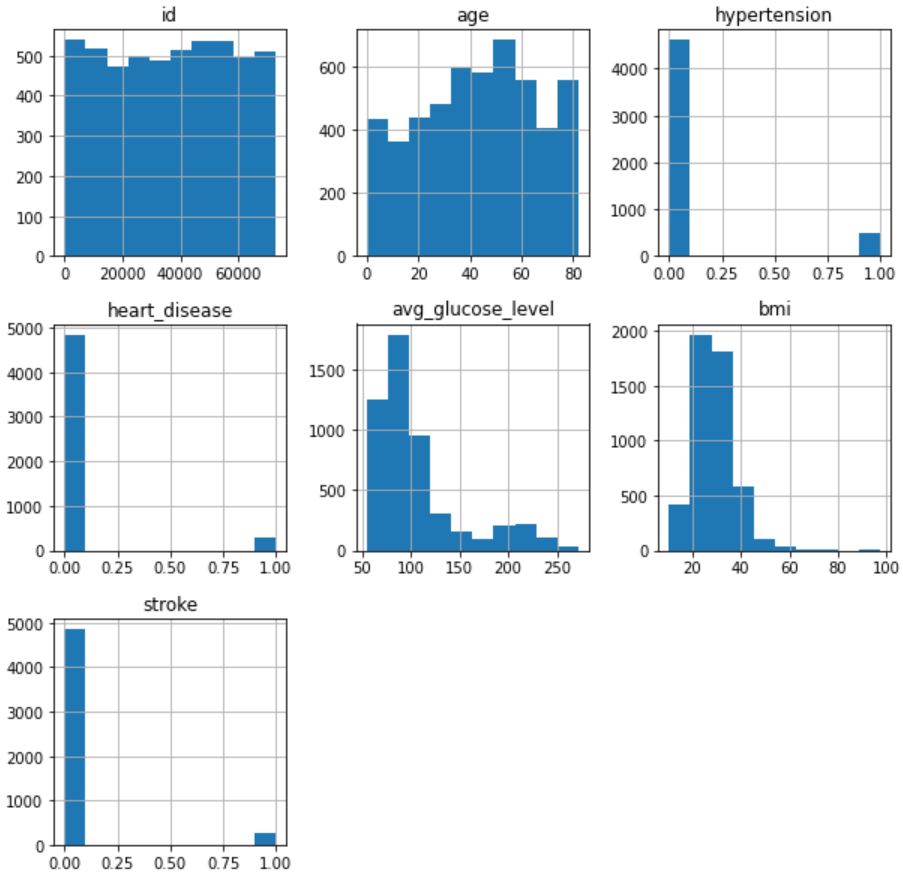
- Các giá trị của siêu tham số:

- clf__criterion: 'gini',
- clf__max_depth: 4,
- clf__max_leaf_nodes: 16,
- clf__min_samples_split: 2

2. Stroke Prediction

a. Mô tả bộ dữ liệu

- Thông tin bộ dữ liệu: Stroke Prediction Dataset
- Jupyter Notebook của nhóm: Stroke_Prediction.ipynb



Hình 7: Mô tả số liệu của các cột trong dữ liệu

- Bộ dữ liệu gồm có 6 trường dữ liệu (trừ cột id) được trực quan hoá ở *Hình 7:*
  - age: tuổi của người khám bệnh.
  - hypertension: chịu chứng tăng huyết áp.
  - heart_disease: có bị bệnh về tim mạch hay không.
  - avg_glucose_level: trung bình lượng đường trong máu.
  - bmi: chỉ số khối của cơ thể.
  - stroke: có bị đột quỵ hay không.

b. Xử lý dữ liệu

- Bước 1: Bỏ cột id và xoá hàng có trường giới tính là Other (vì chỉ có 1 hàng nên không ảnh hưởng đến dữ liệu)

```
df = pd.DataFrame(data)
df = df.dropna()
```

```
df = df.drop(['id'], axis=1) # Bỏ cột id
df = df.drop([3116]) # Xoá hàng có gender là Other
```

- Bước 2: Chuẩn hoá dữ liệu sử dụng **Label Binarizer** và **One Hot Encoding**

```
lb = preprocessing.LabelBinarizer()

X_gender = lb.fit_transform(X_gender)
X_ever_married = lb.fit_transform(X_ever_married)
X_Residence_type = lb.fit_transform(X_Residence_type)

enc = OneHotEncoder(handle_unknown='ignore')

X_work_type = enc.fit_transform(X_work_type).toarray()
X_smoking_status = enc.fit_transform(X_smoking_status).
toarray()
```

- Bước 3: Nối các dữ liệu sau khi chuẩn hoá

```
X_data = np.hstack([X_data, X_gender])
X_data = np.hstack([X_data, X_ever_married])
X_data = np.hstack([X_data, X_work_type])
X_data = np.hstack([X_data, X_Residence_type])
X_data = np.hstack([X_data, X_smoking_status])
```

c. Huấn luyện và đánh giá mô hình

- Vì dữ liệu bị mất cân bằng khá nhiều, nên nhóm đã sử dụng **KNNImputer** của **sklearn** để khắc phục tình trạng này (thể hệ ở biểu đồ stroke ở Hình 7).

```
X_train, X_test, y_train, y_test = train_test_split(X_data,
y_data, test_size=0.33, random_state=42)

imputer = KNNImputer(n_neighbors=2)
X_train = imputer.fit_transform(X_train)
X_test = imputer.fit_transform(X_test)
```

- Không sử dụng Kflod: accuracy 0.91%

```
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(classification_report(y_pred, y_test))
```

	precision	recall	f1-score	support
0	0.95	0.95	0.95	1529
1	0.14	0.12	0.13	91
accuracy			0.91	1620
macro avg	0.54	0.54	0.54	1620
weighted avg	0.90	0.91	0.91	1620

- Sử dụng Kflod: accuracy 0.92%

```
kfold = KFold(n_splits=5, shuffle=True, random_state=1)
accuracy_list = []
for train, test in kfold.split(X_data, y_data):
    clf = DecisionTreeClassifier()

    X_train, X_test = X_data[train], X_data[test];
    y_train, y_test = y_data[train], y_data[test];

    X_train = imputer.fit_transform(X_train)
    X_test = imputer.fit_transform(X_test)

    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy_list.append(accuracy_score(y_test, y_pred))
print("Accuracy: ", np.mean(accuracy_list) * 100)
```

- Nhìn kết quả ở trên, ta có thể thấy mô hình đã bị **overfit** mặc dù nhóm đã sự dụng **KNNImputer** để khắc phục thì trạng này.
- Theo suy nghĩ của nhóm thì mô hình **overfit** thì không có gì quá bất ngờ, vì sự chênh lệch của có đột quỵ (lớp 1) và không bị đột quỵ (lớp 0) là quá cao.

d. Điều chỉnh siêu tham số cho mô hình

```
X_train, X_test, y_train, y_test =
train_test_split(X_data, y_data, test_size=0.33,
random_state=42)
X_train = imputer.fit_transform(X_train)
X_test = imputer.fit_transform(X_test)

parameters = {
    'clf__max_depth': [2, 3, 4],
    'clf__criterion': ['gini', 'entropy'],
    'clf__min_samples_split': [2, 4],
```

```
    'clf__max_leaf_nodes': [8, 16, 32],
}

pipeline = Pipeline(
    steps=[("clf", DecisionTreeClassifier())]
)

gscv = GridSearchCV(pipeline, parameters, cv=5,
n_jobs=12, scoring='accuracy', return_train_score=True,
error_score=0, verbose=3)
gscv.fit(X_train, y_train)
```

	precision	recall	f1-score	support
0	1.00	0.95	0.97	1620
1	0.00	0.00	0.00	0
accuracy			0.95	1620
macro avg	0.50	0.48	0.49	1620
weighted avg	1.00	0.95	0.97	1620

- Nhìn hình trên ta có thể thấy được việc tunning tham số cũng không khắc phục được tình trạng overfitting này.

CHƯƠNG 5: KẾT LUẬN

	Cardiovascular Disease Predict	Stroke Prediction
Normal	65%	91%
Kflod	63,95%	92%
Tunning Hyperparameter	73%	95%

- Nhóm đã thử so sánh với kết quả của nhóm với những người khác trên Kaggle thì thấy rằng độ chính xác chệch lệch không nhiều.Vì vậy, nhìn chung thuật toán **Decision Tree Classifier** là thuật toán phân loại khá tốt đối với bộ dữ liệu cân bằng.
- Tuy nhiên, thuật toán này rất dễ rơi vào tình trạng **overfitting** thể hiện rõ ở bộ dữ liệu dự đoán đột quỵ mặc dùng đã tunning tham số nhưng kết quả vẫn không được cải thiệt. Điều này đưa đến kết luận rằng thuật toán **Decision Tree Classifier** không yêu cầu quá nhiều trong việc tunning.

CHƯƠNG 6: TÀI LIỆU THAM KHẢO

https://www.youtube.com/watch?v=ZVR2Way4nwQ

https://trituenhantao.io/kien-thuc/decision-tree/

https://scikit-learn.org/stable/modules/tree.html

https://machinelearningcoban.com/2018/01/14/id3/