from PyQt6 import QtCore, QtGui, QtWidgets

from PyQt6.QtWidgets import  QTableWidget, QTableWidgetItem

from matplotlib.backends.backend\_qt5agg import FigureCanvasQTAgg as FigureCanvas

from sklearn.metrics import roc\_curve, auc

from matplotlib.legend\_handler import HandlerLine2D

from sklearn import tree

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sb

from sklearn import preprocessing

##### Bước : Thu thập dữ liệu đầu vào

df = pd.read\_csv("D:/HeHoTroQuyetDinh/LungCancer/survey\_lung\_cancer.csv")

print("\nNhận dữ liệu đầu vào:")

print(df)

print("Mô tả thống kê dữ liệu đầu vào:")

print(df.describe())

### Các trường triệu chứng bệnh đều ảnh hướng đến bệnh nhân có bị ung thư hay không nên không cần bước xóa cột không được sủ dụng trong phân lớp###

# Kiểm tra có ô nào có giá trị null hay not value hay không

sb.heatmap(df.isnull())

#print(sb.heatmap(df.isnull()))

##### Bước: Tiền sử lý dữ liệu

#Đọc thông tin toàn bộ bảng dữ liệu để thấy kiểu dữ liệu của từng trường, trường nào kiểu string,object thì phải chuyển về số

print(df.info())

# thực hiện quá trình mã hóa nhãn chuyển đổi chúng thành các giá trị số tương ứng

# các trường có dữ liệu 1-2 chuyển thành 0-1

# trường GENDER : MALE-FEMALE chuyển thành 0-1

# trường LUNG\_CANCER : NO-YES chuyển thành 0-1

le=preprocessing.LabelEncoder()

df['GENDER']=le.fit\_transform(df['GENDER'])

df['AGE']=le.fit\_transform(df['AGE'])

df['SMOKING']=le.fit\_transform(df['SMOKING'])

df['YELLOW\_FINGERS']=le.fit\_transform(df['YELLOW\_FINGERS'])

df['ANXIETY']=le.fit\_transform(df['ANXIETY'])

df['PEER\_PRESSURE']=le.fit\_transform(df['PEER\_PRESSURE'])

df['CHRONIC DISEASE']=le.fit\_transform(df['CHRONIC DISEASE'])

df['FATIGUE ']=le.fit\_transform(df['FATIGUE '])

df['ALLERGY ']=le.fit\_transform(df['ALLERGY '])

df['WHEEZING']=le.fit\_transform(df['WHEEZING'])

df['ALCOHOL CONSUMING']=le.fit\_transform(df['ALCOHOL CONSUMING'])

df['COUGHING']=le.fit\_transform(df['COUGHING'])

df['SHORTNESS OF BREATH']=le.fit\_transform(df['SHORTNESS OF BREATH'])

df['SWALLOWING DIFFICULTY']=le.fit\_transform(df['SWALLOWING DIFFICULTY'])

df['CHEST PAIN']=le.fit\_transform(df['CHEST PAIN'])

df['LUNG\_CANCER']=le.fit\_transform(df['LUNG\_CANCER'])

# trường AGE có kiểu số nhưng có giá trị quá lớn so với các trường khác làm chi phối đến toàn bộ kết quả nên cũng phải chuyển giấ trị từ 0->1

Age=df['AGE']

Age=Age.values.reshape(-1,1)

min\_max\_scaler=preprocessing.MinMaxScaler()

Age\_scaled=min\_max\_scaler.fit\_transform(Age)

df['AGE']=Age\_scaled

print("Tập dữ liệu sau khi chuyển đổi :")

print(df)

##### Bước : Chia tập dữ liệu thành 2 tệp, tệp dữ liệu huấn luyện và tệp dữ liệu thử nghiệm

inputs=df.drop('LUNG\_CANCER',axis="columns") # các cột thuộc tính triệu chứng gây ảnh hưởng tới lung\_cancer - tập vào

target=df['LUNG\_CANCER'] # là kết quả phân hoạch -  tập đích

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(inputs,target,test\_size=0.3,random\_state=50)

##### Bước: Xây dựng mô hình rừng cây từ tập dữ liệu huấn luyện

from sklearn.ensemble import RandomForestClassifier

# n\_estimators(số lượng cây), max\_features(số lượng đặc trưng nhiều nhất trong 1 cây), max\_depth(độ sâu của cây),

# min\_samples\_split(số lượng mẫu tối thiểu để chia), min\_samples\_leaf(số lượng mẫu tối thiểu trong lá),...

clf=RandomForestClassifier(n\_estimators=300,max\_features=15,min\_samples\_split=3,max\_depth=16,random\_state=0)

##### Bước: Huấn luyện mô hình

clf.fit(X\_train,Y\_train)  # Huấn luyện

Y\_pred = clf.predict(X\_test)# Dự đoán nhãn trên tập thử nghiệm

clf.score(X\_test,Y\_test) # Độ chính xác của mô hình

print("Độ chính xác của mô hình :")

print(clf.score(X\_test,Y\_test))

print("Dự đoán với mẫu dữ liệu đầu vào :")

print(Y\_pred)

###Mức độ quan trọng của các thuộc tính

inputsss=df.drop('LUNG\_CANCER',axis="columns")

feature\_imp = pd.Series(clf.feature\_importances\_,index=inputsss.columns).sort\_values(ascending=False)

print("\nMức độ quan trọng của các thuộc tính:")

print(feature\_imp)

##### Bước: đánh giá mô hình

from sklearn.metrics import accuracy\_score , precision\_score, recall\_score

accuracy = accuracy\_score(Y\_pred,Y\_test)

precision = precision\_score(Y\_pred,Y\_test)

recall = recall\_score(Y\_pred,Y\_test)

print("\nĐánh giá mô hình :")

print("Accuracy:", accuracy)

print("Precision:", precision)

print("Recall:", recall)

# đồ thị AUC(Area Under the Curve) biểu diễn mức độ phân loại của mô hình. truc x với các giá trị khác nhau cho tham số n\_estimators là số lượng cây trong mô hình

# Trục y biểu thị giá trị AUC, đo lường khả năng phân loại của mô hình. đường cong màu xanh thể hiện giá trị AUC trên tập dữ liệu huấn luyện (X\_train, Y\_train)

# đường cong màu đỏ thể hiện giá trị AUC trên tập dữ liệu kiểm tra (X\_test, Y\_test)

# Nếu đường cong Train AUC và Test AUC gần nhau và tiến gần đến 1, điều đó cho thấy mô hình có khả năng phân loại tốt.

class show\_auc(FigureCanvas):

    def \_\_init\_\_(self):

        self.fig, self.ax=plt.subplots()

        super().\_\_init\_\_(self.fig)

        plt.close()

        plt.ion()

        n\_estimators = [1,2,4,8,16,32,64,128,300]

        train\_results=[]

        test\_results=[]

        for estimator in n\_estimators:

            rf=RandomForestClassifier(n\_estimators=estimator,n\_jobs=-1)

            rf.fit(X\_train,Y\_train)

            train\_pred=rf.predict(X\_train)

            false\_positive\_rate, true\_positive\_rate,thresholds = roc\_curve(Y\_train,train\_pred)

            roc\_auc =auc(false\_positive\_rate,true\_positive\_rate)

            train\_results.append(roc\_auc)

            y\_pred=rf.predict(X\_test)

            false\_positive\_rate, true\_positive\_rate,thresholds = roc\_curve(Y\_test,y\_pred)

            roc\_auc =auc(false\_positive\_rate,true\_positive\_rate)

            test\_results.append(roc\_auc)

        line1, =self.ax.plot(n\_estimators,train\_results,"b",label="Train AUC")

        line2, =self.ax.plot(n\_estimators,test\_results,"r",label="Test AUC")

        self.fig.legend(handler\_map={line1:HandlerLine2D(numpoints=2)})

        self.ax.set\_ylabel("auc score")

        self.ax.set\_xlabel("n\_estimators")

        self.fig.suptitle(' Area Under The Curve (AUC)',size=8)

# ROC là đường cong biểu diễn mức độ phân loại của mô hình

# được vẽ bằng cách biểu diễn tỷ lệ giữa tỷ lệ dương tính thực sự (TPR) và tỷ lệ dương tính giả (FPR) của một hệ thống phân loại

# TPR là tỷ lệ các trường hợp dương tính được phát hiện chính xác trong tổng số các trường hợp dương tính, trong khi FPR là tỷ lệ các trường hợp

# âm tính bị sai lầm được dự đoán là dương tính

# Giá trị roc\_auc là diện tích nằm dưới đường cong ROC. Nó thể hiện khả năng phân loại của mô hình dự đoán dựa trên tỷ lệ TPR và FPR.

class show\_roc(FigureCanvas):

    def \_\_init\_\_(self):

        self.fig, self.ax=plt.subplots()

        super().\_\_init\_\_(self.fig)

        plt.close()

        plt.ion()

        false\_positive\_rate, true\_positive\_rate,thresholds = roc\_curve(Y\_test,Y\_pred)

        roc\_auc =auc(false\_positive\_rate,true\_positive\_rate)

        self.ax.plot(false\_positive\_rate, true\_positive\_rate, label='ROC Curve (AUC = {:.2f})'.format(roc\_auc))

        self.ax.plot([0, 1], [0, 1], 'k--', label='Random Guess')

        self.ax.set\_xlabel('False Positive Rate')

        self.ax.set\_ylabel('True Positive Rate')

        self.fig.suptitle('Receiver Operating Characteristic (ROC)',size=8)

        self.fig.legend(loc='lower right')