Trường Đại học Khoa học Tự nhiên, ĐHQG-HCM

Khoa Công nghệ Thông tin

---o0o---

Đồ Án 2 – Triển Khai Mô Hình Phân Loại Bệnh Ung Thư Vú

Môn: Khai Thác Dữ Liệu Và Ứng Dụng

Giảng viên lý thuyết: - Lê Ngọc Thành

Giảng viên thực hành: - Nguyễn Thái Vũ

Sinh Viên:

- **Ngô Quốc Phát**
- **Nguyễn Trung Hiếu**
- **Trần Xuân Phước**

Mục Lục

I. Tổng quan:

1. Thông tin nhóm:

STT	MSSV	Họ và Tên
1	19127430	Nguyễn Trung Hiếu
2	19127516	Trần Xuân Phước
3	19127503	Ngô Quốc Phát

2. Đánh giá mức độ hoàn thành của mỗi yêu cầu:

STT	Yêu cầu	Hoàn thành
1	Xây dựng ứng dụng gồm 2 chức năng : - Phân tích dữ liệu - Triển khai mô hình phân lớp	100%
2	Báo cáo quá trình thực hiện mô hình phân lớp - Chuẩn bị dữ liệu - Huấn luyện mô hình - Báo cáo kết quả, độ chính xác	100%

3. Mức độ hoàn thành của từng thành viên:

MSSV	Họ và Tên	Nhiệm vụ	Hoàn thành
19127503	Ngô Quốc Phát	- Tiền xử lí và xây dựng mô hình học máy - Hoàn thành báo cáo	100%
19127403	Nguyễn Trung Hiếu	- Tiền xử lí và xây dựng mô hình học máy	100%
19127516	Trần Xuân Phước	- Trực quan hóa dữ liệu - Xây dựng và hoàn thành ứng dụng.	100%

II. Khám phá dữ liệu:

1. Mô tả dữ liệu:

Các đặc điểm được tính toán từ hình ảnh số hóa của một kim hút nhỏ (FNA) của khối vú. Chúng mô tả các đặc điểm của nhân tế bào có trong hình ảnh.

Trong không gian 3 chiều được mô tả trong: [K. P. Bennett và O. L. Mangasarian: "Phân biệt lập trình tuyến tính mạnh mẽ của hai tập hợp tuyến tính không thể tách rời", Phương pháp và phần mềm tối ưu hóa 1, 1992, 23-34].

2. Dữ liệu có bao nhiêu dòng và cột?

Dữ liệu bao gồm 569 dòng và 33 cột.

```
In [5]: df.shape
Out[5]: (569, 33)
```

0	id	569 non-null	int64
1	diagnosis	569 non-null	object
2	radius_mean	569 non-null	float64
3	texture_mean	569 non-null	float64
4	perimeter_mean	569 non-null	float64
5	area_mean	569 non-null	float64
6	smoothness_mean	569 non-null	float64
7	compactness_mean	569 non-null	float64
8	concavity_mean	569 non-null	float64
9	concave points_mean	569 non-null	float64
10	symmetry_mean	569 non-null	float64
11	fractal_dimension_mean	569 non-null	float64
12	radius_se	569 non-null	float64
13	texture_se	569 non-null	float64
14	perimeter_se	569 non-null	float64
15	area_se	569 non-null	float64
16	smoothness_se	569 non-null	float64
17	compactness_se	569 non-null	float64
18	concavity_se	569 non-null	float64
19	concave points_se	569 non-null	float64
20	symmetry_se	569 non-null	float64
21	fractal_dimension_se	569 non-null	float64
22	radius_worst	569 non-null	float64
23	texture_worst	569 non-null	float64
24	perimeter_worst	569 non-null	float64
25	area_worst	569 non-null	float64
26	smoothness_worst	569 non-null	float64
27	compactness_worst	569 non-null	float64
28	concavity_worst	569 non-null	float64
29	concave points_worst	569 non-null	float64
30	symmetry_worst	569 non-null	float64
31	fractal_dimension_worst	569 non-null	float64
32	Unnamed: 32	0 non-null	float64

3. **Thông tin thuộc tính:**

1) Số ID

2) Chẩn đoán (M = ác tính, B = lành tính)

3-32)

Mười đặc điểm có giá trị thực được tính toán cho mỗi nhân tế bào:

a) radius :trung bình của khoảng cách từ tâm đến các điểm trên chu vi)

b) texture :độ lệch chuẩn của các giá trị thang xám

c) perimeter: chu vi

d) area: diện tích

e) smoothness (sự thay đổi cục bộ của chiều dài bán kính)

f) compactness : độ chặt (chu vi ^ 2 / diện tích - 1,0)

g) concavity: độ cong (mức độ nghiêm trọng của các phần lõm của đường viền)

h) concave ponint: điểm lõm (số phần lõm của đường bao)

i) symmetry: đối xứng

j) fractal dimension ("ước lượng đường bờ biển" - 1)

Sai số trung bình, sai số tiêu chuẩn và "lỗi tồi tệ nhất" hoặc lớn nhất (giá trị trung bình trong ba lỗi giá trị lớn nhất) trong số các tính năng này đã được tính toán cho mỗi hình ảnh, dẫn đến 30 tính năng.

- Mean: sai số trung bình
- Se: sai số tiêu chuẩn
- Worst: sai số tệ nhất

4. **Có dữ liệu nào bị trống hay không?**

```
In [10]: df.isna().sum()

Out[10]: diagnosis                0
         radius_mean              0
         texture_mean             0
         perimeter_mean           0
         area_mean                0
         smoothness_mean          0
         compactness_mean         0
         concavity_mean           0
         concave points_mean      0
         symmetry_mean            0
         fractal_dimension_mean   0
         radius_se                0
         texture_se               0
         perimeter_se             0
         area_se                  0
         smoothness_se            0
         compactness_se           0
         concavity_se             0
         concave points_se        0
         symmetry_se              0
         fractal_dimension_se     0
         radius_worst             0
         texture_worst            0
         perimeter_worst          0
         area_worst               0
```

Ta thấy được không có dòng nào bị null.

5. **Các cột đang có kiểu dữ liệu là gì, có cột nào cần hải xử lý dữ liệu không?**

```
0   id                      569 non-null    int64
1   diagnosis               569 non-null    object
2   radius_mean             569 non-null    float64
3   texture_mean            569 non-null    float64
4   perimeter_mean          569 non-null    float64
5   area_mean               569 non-null    float64
6   smoothness_mean         569 non-null    float64
7   compactness_mean        569 non-null    float64
8   concavity_mean          569 non-null    float64
9   concave points_mean     569 non-null    float64
10  symmetry_mean           569 non-null    float64
11  fractal_dimension_mean  569 non-null    float64
12  radius_se               569 non-null    float64
13  texture_se              569 non-null    float64
14  perimeter_se            569 non-null    float64
15  area_se                 569 non-null    float64
16  smoothness_se           569 non-null    float64
17  compactness_se          569 non-null    float64
18  concavity_se            569 non-null    float64
19  concave points_se       569 non-null    float64
20  symmetry_se             569 non-null    float64
21  fractal_dimension_se    569 non-null    float64
22  radius_worst            569 non-null    float64
23  texture_worst           569 non-null    float64
24  perimeter_worst         569 non-null    float64
25  area_worst              569 non-null    float64
26  smoothness_worst        569 non-null    float64
27  compactness_worst       569 non-null    float64
28  concavity_worst         569 non-null    float64
```

- Ta thấy được chỉ có cột "diagnosis" có kiểu dữ liệu là object.
- Mà qua kiểm tra ta thấy kiểu dữ liệu này chỉ có 2 biến là M và B nên ta sẽ chuyển dữ liệu của cột về dạng số với M là 1 và B là 0

```
In [5]:  df.diagnosis.unique()
Out[5]:  array(['M', 'B'], dtype=object)
```

```
In [6]:  df['diagnosis'] = df['diagnosis'].apply(lambda val: 1 if val == 'M' else 0)
```

6. Các thống kê cơ bản của mỗi thuộc tính:

```
In [8]: df.describe()
```

Out[8]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	s
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	
mean	0.372583	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	
std	0.483918	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	
min	0.000000	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	
25%	0.000000	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	
50%	0.000000	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	
75%	1.000000	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	
max	1.000000	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	

8 rows × 31 columns

III. Tiền xử lý dữ liệu:

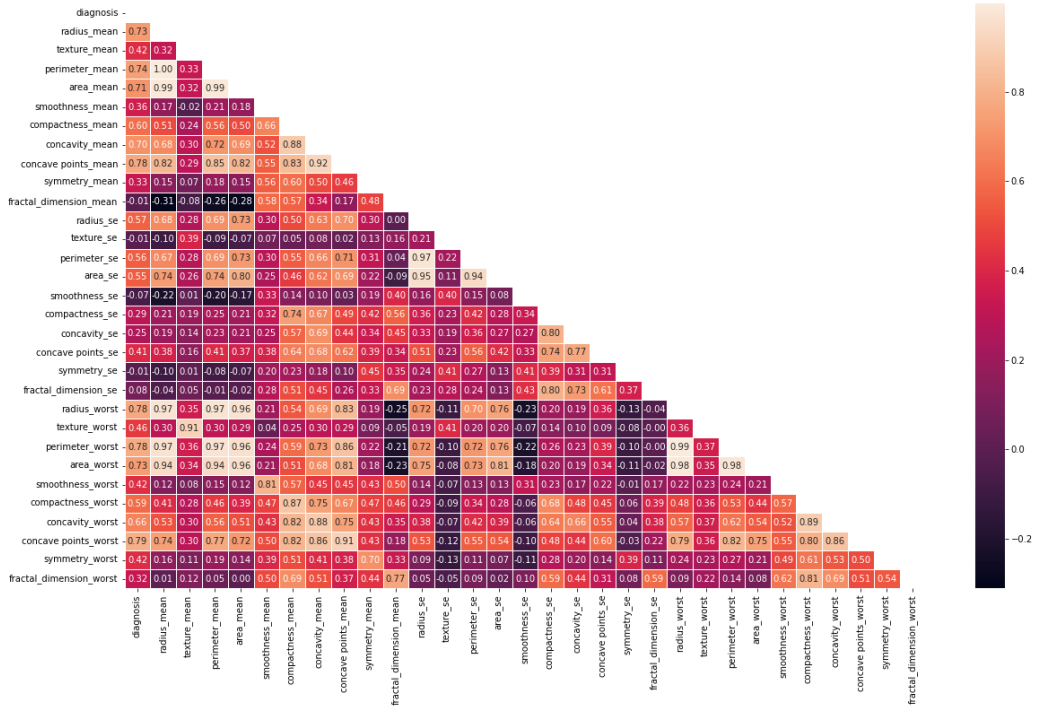
1. Xóa cột Id và unnamed:32

Bởi vì 2 cột này có không cần thiết đến dữ liệu nên ta sẽ tiến hành xóa 2 cột này đi.

```
df.drop(['id', 'Unnamed: 32'], axis = 1, inplace = True)
```

2. Các dữ liệu có mối tương quan với nhau nhiều hay không?

Để tìm hiểu về sự tương quan, ta sẽ vẽ heatmap cho toàn bộ dữ liệu để tìm sự tương quan.



Qua heatmap, ta thấy được có rất nhiều dữ liệu có sự tương quan rất lớn với nhau với chỉ số tương quan vượt 0.9.

Với những cột có hệ số tương quan cao như vậy, ta nên xử lý bằng các xóa dần nó đi để sau khi chạy thực toán, mô hình sẽ đạt được độ chính xác cao hơn. Nếu giữ lại các cột này sẽ ảnh hưởng đến mô hình và kết quả sau khi train.

```
In [32]: corr_matrix = df.corr().abs()

         mask = np.triu(np.ones_like(corr_matrix, dtype = bool))
         tri_df = corr_matrix.mask(mask)

         to_drop = [x for x in tri_df.columns if any(tri_df[x] > 0.9)]

         df = df.drop(to_drop, axis = 1)

         print(f"Số cột còn lại là {df.shape[1]} cột")

         Số cột còn lại là 21 cột
```

IV. Mô hình học máy:

1. Chia tập huấn luyện, kiểm thử:

- Chia thành tập x_train, y_train, x_test, y_test
- Tập train chứa 70% dữ liệu, tập test chứa 30% dữ liệu.

```
In [13]: X = df.drop('diagnosis', axis = 1)
         y = df['diagnosis']
```

```
In [14]: 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)
```

2. Sử dụng thư viện StandardScaler để chuẩn dữ liệu:

Để tránh tình trạng quá tập trung vào dữ liệu của một cột nên ta sẽ chuẩn hóa dữ liệu.

```
:  from sklearn.preprocessing import StandardScaler

   scaler = StandardScaler()

   X_train = scaler.fit_transform(X_train)
   X_test = scaler.transform(X_test)
```

3. Thiết lập và train mô hình:

```
In [37]: from sklearn.neighbors import KNeighborsClassifier

         knn = KNeighborsClassifier(n_neighbors=5)
         knn.fit(X_train, y_train)

Out[37]: KNeighborsClassifier()

In [38]: y_pred = knn.predict(X_test)
```

4. Kết quả sau khi train

- **Accuracy score (độ chính xác)**

```
In [39]: from sklearn.metrics import accuracy_score
         print(accuracy_score(y_train, knn.predict(X_train)))

         knn_acc = accuracy_score(y_test, knn.predict(X_test))
         print(knn_acc)

         0.9522613065326633
         0.935672514619883
```

Ta thấy được khi sử dụng mô hình vừa train được với tập train, ta được độ chính xác là 95.2% , và khi dung với tập test thì ta được độ chính xác là 93.5 %

- **Confusion matrix:**

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_pred))
```

```
[[104   4]
 [  7  56]]
```

Ta thấy khả năng dự đoán của mô hình là rất tốt với tỉ lệ dự đoán sai là rất thấp.

- 104 lần dự đoán đúng bệnh nhân ở giai đoạn B
- 56 lầ dự đoán đúng bệnh nhận ở giai đoạn M
- 7 lần dự đoán sai bệnh nhân ở giai đoạn B
- 4 lần dự đoán sai bệnh nhân ở giai đoạn M
- **Classification report:**

```
In [22]: from sklearn.metrics import classification_report
         print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.94	0.96	0.95	108
1	0.93	0.89	0.91	63
accuracy			0.94	171
macro avg	0.94	0.93	0.93	171
weighted avg	0.94	0.94	0.94	171

Precision: cho biết độ chính xác của quá trình phân lớp đối với từng loại nhãn.

- Với bệnh nhân loại B thì có độ chính xác là 94%
- Với bệnh nhân loại M thì có độ cính xác là 93%

V. Xây dựng ứng dụng phân tích dữ liệu và dự đoán bệnh ung thư:

1. Các thư viện được sử dụng:

```
import tkinter.messagebox
from tkinter import ttk
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import seaborn as sns
import pandas as pd
import numpy as np
from tkinter import *
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from tkinter import filedialog
from sklearn.preprocessing import MinMaxScaler
```

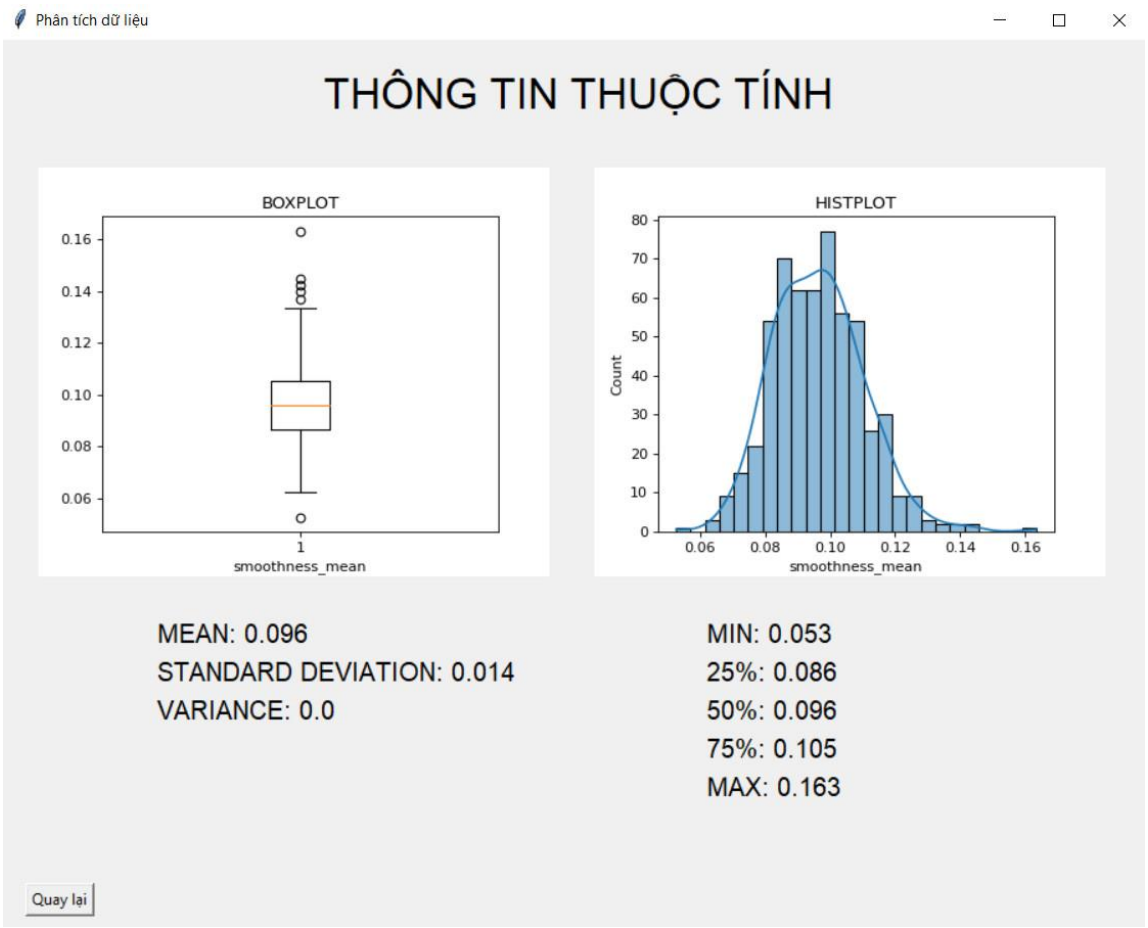
2. Phân tích dữ liệu và vẽ biểu đồ:

```
def VisualScreen(index_col):
    name_col = name_column[index_col]
    visualScreen = Tk()
    visualScreen.title('Phân tích dữ liệu')
    visualScreen.geometry('900x700')
    Label(visualScreen, text='THÔNG TIN THUỘC TÍNH', font=('bold', 25)).place(x=250, y=20)

    # Thông tin thuộc tính: MEAN, STD, VAR, Min, Max,...
    Label(visualScreen, text='MEAN: '+str(round(df[name_col].mean(), 3)), font=('bold', 15)).place(x=120, y=450)
    Label(visualScreen, text='STANDARD DEVIATION: '+str(round(df[name_col].std(),3)), font=('bold', 15)).place(x=120, y=480)
    Label(visualScreen, text='VARIANCE: '+str(round(df[name_col].var(), 3)), font=('bold', 15)).place(x=120, y=510)

    cal_values = np.percentile(df[name_col], q=[0, 25, 50, 75, 100])

    Label(visualScreen, text='MIN: '+str(round(cal_values[0], 3)), font=('bold', 15)).place(x=550, y=450)
    Label(visualScreen, text='25%: '+str(round(cal_values[1], 3)), font=('bold', 15)).place(x=550, y=480)
    Label(visualScreen, text='50%: '+str(round(cal_values[2], 3)), font=('bold', 15)).place(x=550, y=510)
    Label(visualScreen, text='75%: '+str(round(cal_values[3], 3)), font=('bold', 15)).place(x=550, y=540)
    Label(visualScreen, text='MAX: '+str(round(cal_values[4], 3)), font=('bold', 15)).place(x=550, y=570)

    # BOXPLOT của thuộc tính được chọn
```

```
# BOXPLOT của thuộc tính được chọn
figure1 = plt.Figure(figsize=(5, 4), dpi=80)
ax1 = figure1.add_subplot(111)
ax1.boxplot(df[name_col])
boxplot1 = FigureCanvasTkAgg(figure1, visualScreen)
boxplot1.get_tk_widget().place(x=30, y=100)
ax1.set_xlabel(name_column[index_col])
ax1.set_title('BOXPLOT')
# HISPLOT của thuộc tính được chọn
figure2 = plt.Figure(figsize=(5, 4), dpi=80)
ax2 = figure2.add_subplot(111)
sns.histplot(df[name_col], kde=True, ax=ax2)
hisplot2 = FigureCanvasTkAgg(figure2, visualScreen)
hisplot2.get_tk_widget().place(x=465, y=100)
ax2.set_xlabel(name_column[index_col])
ax2.set_title('HISTPLOT')

Button(visualScreen, text='Quay lại', command=visualScreen.destroy).place(x=20, y=660)

visualScreen.mainloop()
```

Để có thể nhìn nhận nhanh về dữ liệu, nhóm sử dụng 2 loại biểu đồ là boxplot và distplot.

- Boxplot: Boxplot thể hiện phân phối dữ liệu của các thuộc tính số thông qua các “tứ phân vị” Một biểu đồ boxplots chia tập dữ liệu thành những khoảng phần tư (quartiles).
- Distplot: Thể hiện sự phân bố của dữ liệu.

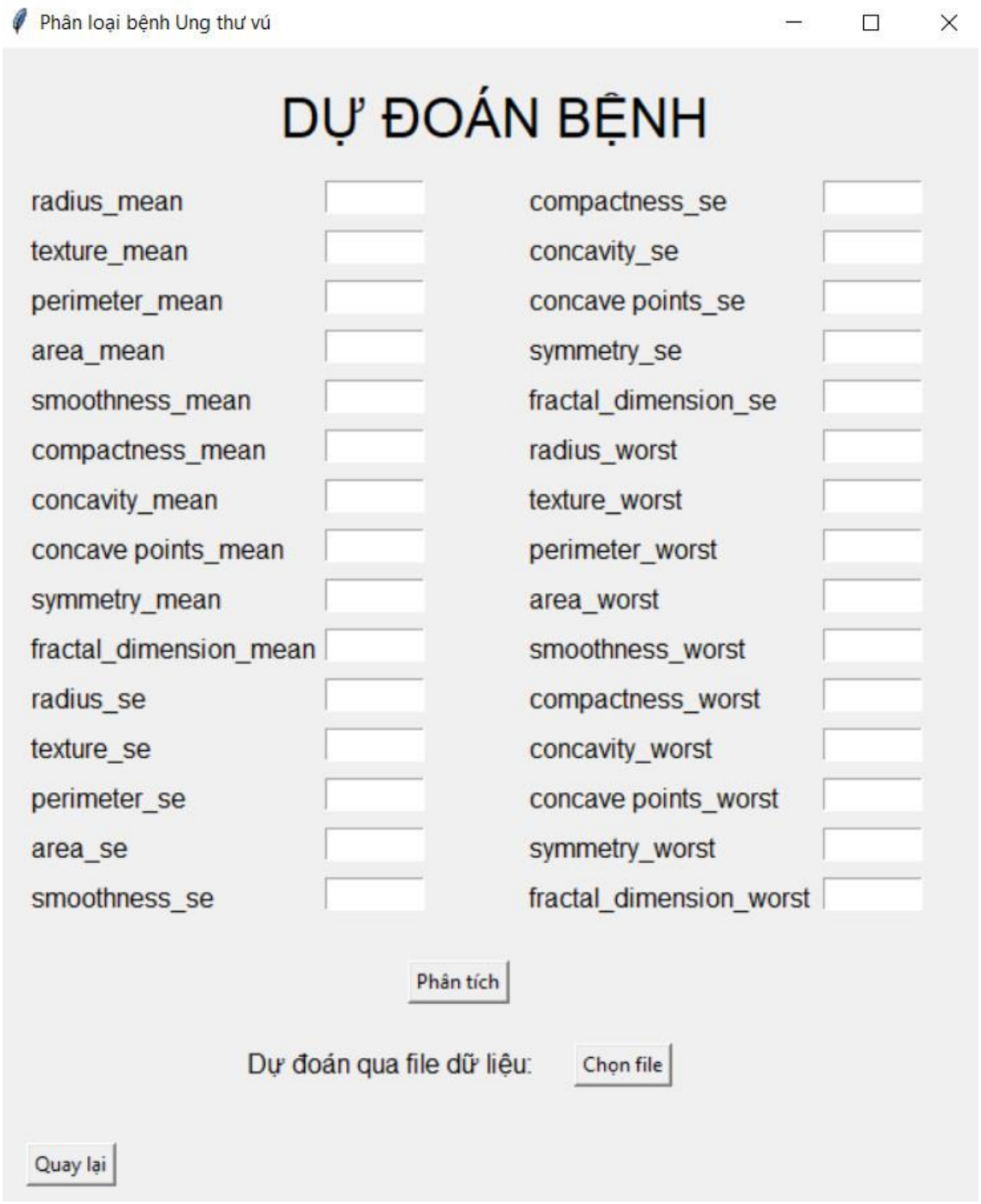


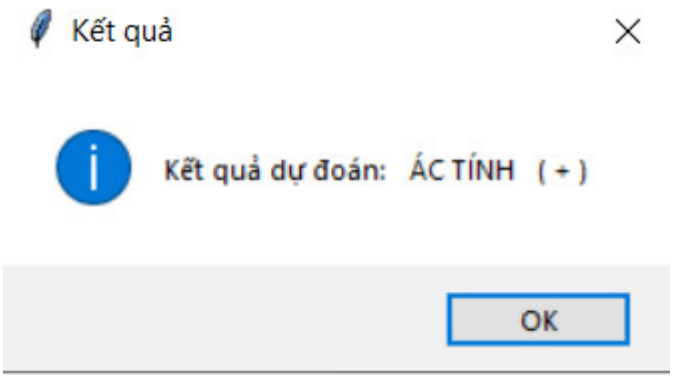
3. Triển khai mô hình phân lớp:

```
# Đọc dữ liệu để huấn luyện
df = pd.read_csv('data.csv')
name_column = df.drop(columns=['diagnosis', 'id', 'Unnamed: 32'], axis=1).columns
df.drop(['id','Unnamed: 32'], axis=1, inplace=True)
knn = KNeighborsClassifier(n_neighbors=10)
scaler = MinMaxScaler()

def ML(df):
    corr_matrix = df.corr().abs()
    mask = np.triu(np.ones_like(corr_matrix, dtype = bool))
    tri_df = corr_matrix.mask(mask)
    # Xóa những thuộc tính có độ tương quan cao
    global to_drop
    to_drop = [x for x in tri_df.columns if any(tri_df[x] > 0.9)]
    df = df.drop(to_drop, axis = 1)

    y = df['diagnosis']
    X = df.drop('diagnosis',axis=1)

    # chia tập huấn luyện và tập kiểm tra
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 0)

    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    # huấn luyện
    knn.fit(X_train, y_train)
```

```
def analy_data():
    # Lấy thông tin mới được nhập bởi người dùng
    for i in range(30):
        value = data_input[i].get("1.0","end-1c")
        if value != '':
            data_input[i] = value
        else:
            data_input[i] = '0'
    data = pd.DataFrame([data_input], columns=name_column).astype('float')
    global to_drop
    data = data.drop(to_drop, axis=1)
    data = scaler.transform(data)
    # Dự đoán kết quả
    result = knn.predict(data)
    if result == "['B']":
        tkinter.messagebox.showinfo(title="Kết quả", message=' Kết quả dự đoán:   LÀNH TÍNH   ( - ) ')
    else:
        tkinter.messagebox.showinfo(title="Kết quả", message=' Kết quả dự đoán:   ÁC TÍNH   ( + ) ')
data_input = [None] * 30
```

Ta sẽ có được một màn hình để người dung có thể nhập các dữ liệu và xuất ra kết quả dự đoán.


Phân loại bệnh Ung thư vú
DỰ ĐOÁN BỆNH
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave points_mean
symmetry_mean
fractal_dimension_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave points_se
symmetry_se
fractal_dimension_se
radius_worst
texture_worst
perimeter_worst
area_worst
smoothness_worst
compactness_worst
concavity_worst
concave points_worst
symmetry_worst
fractal_dimension_worst
Phân tích
Dự đoán qua file dữ liệu:
Chọn file
Quay lại



- Ứng dụng còn có tính năng người dung import 1 file csv chứa các dữ liệu để có thể dự đoán. Tính năng này giúp người dung tiện cho việc sai sót trong khi nhập dữ liệu và cũng như có thể làm nhiều dữ liệu cùng 1 lúc.

VI. Tài liệu tham khảo:

+ https://pythonguides.com/python-tkinter-multiple-windows-tutorial/

+ https://www.pythontutorial.net/tkinter/tkinter-toplevel/

+ https://www.geeksforgeeks.org/how-to-embed-matplotlib-charts-in-tkinter-gui/

+ https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

+ https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

+ https://www.kaggle.com/code/gargmanish/basic-machine-learning-with-cancer