**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP.HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**A logo with a book and a graduation cap

Description automatically generated-----o0o----**

**Báo cáo đồ án:**

**Xây dựng ứng dụng dự đoán bệnh thận bằng mô hình Logistic Regresstion**

**Giảng viên hướng dẫn: Trần Đình Toàn**

**Nhóm 18 (Sunrise)**

**Thành viên:**

**1: Vũ Thị Bảo Yến - 2001216334**

**2: Phạm Thị Thanh Tình - 2001216218**

**3: Đinh Hoàng Huy - 2001215809**

**Thành phố Hồ Chí Minh, tháng 4 năm 2024**

**Lời Cam Đoan**

Chúng em xin cam đoan đề tài chủ đề: Logistic Regression và ứng dụng dự đoán bệnh thận là do nhóm tự tìm hiểu

Chúng em đã kiểm tra dữ liệu theo quy định hiện hành.

Kết quả bài làm của tìm hiểu Mạng Logistic Regression và ứng dụng dự đoán bệnh thận là trung thực và không sao chép từ bất kỳ bài tập của nhóm khác.

**Lời Cảm Ơn**

Lời đầu tiên, em xin gửi lời cảm ơn chân thành nhất đến thầy Trần Đình Toàn. Trong quá trình học tập và tìm hiểu bộ môn Học máy, em đã nhận được sự quan tâm giúp đỡ, hướng dẫn rất tận tình, tâm huyết của thầy. Thầy đã giúp chúng em tích lũy thêm nhiều kiến thức để có cái nhìn sâu sắc và hoàn thiện hơn trong bộ môn này. Từ những kiến thức mà thầy truyền tải, em đã dần trả lời được những câu hỏi, những thắc mắc của em khi vừa bắt đầu học Học máy. Thông qua bài báo cáo này, nhóm em xin trình bày những gì mà nhóm đã tìm hiểu về hiểu về đề tài “ Logistic Regression và ứng dụng dự đoán bệnh thận ”, cũng như thể hiện ra những kiến thức mà em đã học được từ thầy từ đầu học kỳ đến hiện tại.

Có lẽ kiến thức là vô hạn mà sự tiếp nhận kiến thức của bản thân mỗi người luôn tồn tại những hạn chế nhất định. Do đó, trong quá trình hoàn thành bài báo cáo, chắc chắn không tránh khỏi những thiếu sót. Bản thân em cũng như các thành viên trong nhóm rất mong nhận được những góp ý đến từ thầy để bài báo cáo của nhóm em được hoàn thiện hơn.

Kính chúc thầy nhiều sức khỏe, hạnh phúc và thành công trên con đường sự nghiệp giảng dạy.

**MỤC LỤC**

[CHƯƠNG 1: MỞ ĐẦU 1](#_Toc167736944)

[1.1 Lý do chọn đề tài 1](#_Toc167736945)

[1.2 Mục tiêu đề tài 1](#_Toc167736946)

[CHƯƠNG 2: LOGISTIC REGRESSION (HỒI QUY LOGISTIC) 3](#_Toc167736947)

[2.1 Logistic Regression là gì? 3](#_Toc167736948)

[2.2 Cách hoạt động của Logistic Regression dựa trên hàm Sigmoid 3](#_Toc167736949)

[2.3 Công thức của hàm mất mát Cross-Entropy 4](#_Toc167736950)

[2.4 Các ứng dụng của Logistic Regression 4](#_Toc167736951)

[2.5 Ưu nhược điểm của Logistic Regression 4](#_Toc167736952)

[2.5.1 Ưu điểm 5](#_Toc167736953)

[2.5.2Nhược điểm 5](#_Toc167736954)

[CHƯƠNG 3: XỬ LÝ DỮ LIỆU VÀ TRAIN MODEL 6](#_Toc167736955)

[3.1 Mô tả dataset 6](#_Toc167736956)

[3.2 Các bước thực hiện 6](#_Toc167736957)

[3.2.1 Tiền xử lý dữ liệu 6](#_Toc167736958)

[3.2.2 Train model Logictic Regresstion 15](#_Toc167736959)

[CHƯƠNG 4: XÂY DỰNG ỨNG DỤNG 17](#_Toc167736960)

[4.1 Mô tả ứng dụng 17](#_Toc167736961)

[4.2 Giao diện ứng dụng 17](#_Toc167736962)

[TÀI LIỆU THAM KHẢO 23](#_Toc167736963)

# CHƯƠNG 1: MỞ ĐẦU

## 1.1 Lý do chọn đề tài

Bệnh suy thận là một trong những vấn đề sức khỏe của con người nghiêm trọng nhất trên toàn thế giới, ảnh hưởng đến cuộc sống của các người bệnh và gia đình của họ.Đây là loại bệnh phổ biến nhất ở người già và trung niên, và mặc dù có sự tiến bộ trong việc chẩn đoán và điều trị, nhưng vẫn còn nhiều thách thức cần được giải quyết.

Do đó đề tài này chọn lựa vấn đề này vì sự cần thiết của việc tạo ra các công cụ hỗ trợ chẩn đoán và điều trị bệnh suy thận. Bằng cách áp dụng các thông tin và thuật toán Logistic Regression, đề tài này hy vọng cung cấp một công cụ hữu ích cho các bác sĩ và nhà nghiên cứu y học trong việc phân tích và đưa ra quyết định chẩn đoán và điều trị cho bệnh nhân suy thận. Điều này có thể dẫn đến việc cải thiện chính xác và hiệu quả của quy trình điều trị, cũng như giảm thiểu tác động của bệnh lý đối với bệnh nhân.

## 1.2 Mục tiêu đề tài

Xây dựng và huấn luyện một mô hình machine learning dựa trên thuật toán logistic regression để dự đoán nguy cơ mắc bệnh thận.

Thu thập và chuẩn bị tập dữ liệu về bệnh nhân mắc bệnh thận, bao gồm các yếu tố rủi ro như độ tuổi,huyết áp, trọng lượng riêng, lượng đường, tế bào máu đỏ, lượng urea trong máu,.... Chia tập dữ liệu thành tập huấn luyện và tập kiểm tra.

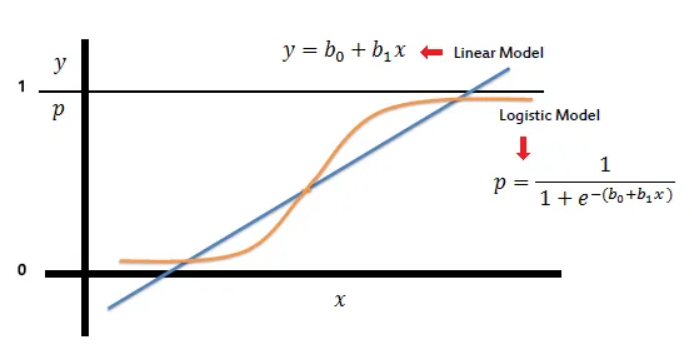
Đánh giá hiệu suất của mô hình dự đoán:Tính các chỉ số hiệu suất trên tập kiểm tra, bao gồm độ chính xác..Đánh giá khả năng tổng quát hóa của mô hình bằng cách kiểm tra trên các tập dữ liệu khác nhau.

Phân tích các yếu tố rủi ro quan trọng:Xác định các yếu tố rủi ro quan trọng nhất ảnh hưởng đến nguy cơ mắc bệnh thận thông qua phân tích hệ số hồi quy của mô hình logistic regression.Tìm hiểu mối quan hệ giữa các yếu tố rủi ro và khả năng mắc bệnh thận.Đánh giá mức độ ảnh hưởng của từng yếu tố rủi ro đến khả năng mắc bệnh.

Ứng dụng mô hình dự đoán trong sàng lọc và theo dõi: Từ việc train model xong và dựa vào feature chúng em sử dụng mô hình để sàng lọc và xác định nhóm người có nguy cơ mắc bệnh thận cao.Để đề xuất các biện pháp can thiệp và theo dõi định kỳ phù hợp cho nhóm có nguy cơ cao, nhằm phát hiện sớm và ngăn ngừa sự tiến triển của bệnh.Đánh giá hiệu quả của các biện pháp can thiệp thông qua theo dõi diễn biến sức khỏe của nhóm có nguy cơ cao.

# CHƯƠNG 2: LOGISTIC REGRESSION (HỒI QUY LOGISTIC)

## 2.1 Logistic Regression là gì?

Hồi quy Logistic là một mô hình thống kê được sử dụng để phân loại nhị phân, tức dự đoán một đối tượng thuộc vào một trong hai nhóm. Hồi quy Logistic làm việc dựa trên nguyên tắc của hàm sigmoid – một hàm phi tuyến tự chuyển đầu vào của nó thành xác suất thuộc về một trong hai lớp nhị phân.

## 2.2 Cách hoạt động của Logistic Regression dựa trên hàm Sigmoid

Hồi quy Logistic hoạt động dựa trên hàm Sigmoid, được biểu diễn như sau:

𝑆(𝑧) =

Hàm Sigmoid nhận đầu vào là một giá trị z bất kỳ, và trả về đầu ra là một giá trị xác suất nằm trong khoảng [0,1]. Khi áp dụng vào mô hình Hồi quy Logistic với đầu vào là ma trận dữ liệu 𝑋 và trọng số w, ta có z=Xw.

Việc huấn luyện của mô hình là tìm ra bộ trọng số 𝑤 sao cho đầu ra dự đoán của hàm Sigmoid gần với kết quả thực tế nhất. Để làm được điều này, ta sử dụng hàm mất mát (Loss Function) để đánh giá hiệu năng của mô hình. Mô hình càng tốt khi hàm mất mát càng nhỏ.

Hàm mất mát (Loss Function) là một hàm số được sử dụng để đo lường mức độ lỗi mà mô hình của chúng ta tạo ra khi dự đoán các kết quả từ dữ liệu đầu vào. Trong bài toán Hồi quy Logistic, chúng ta sử dụng hàm mất mát Cross-Entropy (còn gọi là Log Loss) để đánh giá hiệu năng của mô hình

## 2.3 Công thức của hàm mất mát Cross-Entropy

Hàm mất mát [Cross-Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/cross-entropy/) được định nghĩa như sau:

Trong đó:

* 𝑛: số lượng mẫu dữ liệu trong tập huấn luyện.
* : giá trị thực tế của đầu ra thứ i.
* xác suất dự đoán thuộc lớp 1 của mô hình cho đầu vào thứ i.

Hàm [Cross-Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/cross-entropy/) đo lường khoảng cách giữa hai phân phối xác suất ​ và . Khi mô hình dự đoán chính xác, tức là nếu ​ ​=1 thì ​ càng gần 1, và nếu ​=0 thì ​ càng gần 0, sau đó hàm mất mát sẽ tiến gần về 0.

Trong quá trình huấn luyện, chúng ta tìm cách cập nhật bộ trọng số 𝑤 sao cho giá trị hàm mất mát [Cross-Entropy](https://trituenhantao.io/tu-dien-thuat-ngu/cross-entropy/) đạt giá trị nhỏ nhất, dẫn đến một mô hình dự đoán tốt nhất.

Để tìm giá trị tối ưu cho bộ trọng số w, chúng ta có thể sử dụng kỹ thuật [Gradient](https://trituenhantao.io/tu-dien-thuat-ngu/gradient/) Descent. Tại mỗi bước lặp, chúng ta cập nhật 𝑤 theo phương từm ứng với đạo hàm của hàm mất mát 𝐿(𝑤) theo 𝑤.

## 2.4 Các ứng dụng của Logistic Regression

* Y học: Dự đoán nguy cơ mắc bệnh, phân tích hiệu quả của phương pháp điều trị.
* Kinh doanh: Dự đoán xác suất mua hàng, phân loại khách hàng tiềm năng.
* Chính trị: Dự đoán kết quả bầu cử, phân tích hành vi bỏ phiếu.

## 2.5 Ưu nhược điểm của Logistic Regression

### 2.5.1 Ưu điểm

**Tính đơn giản**

Các mô hình hồi quy logistic ít phức tạp về mặt toán học hơn các phương pháp ML khác. Do đó, bạn có thể triển khai chúng ngay cả khi đội ngũ của bạn không ai có chuyên môn sâu về ML.

**Tốc độ**

Các mô hình hồi quy logistic có thể xử lý khối lượng lớn dữ liệu ở tốc độ cao bởi chúng cần ít khả năng điện toán hơn, chẳng hạn như bộ nhớ và sức mạnh xử lý. Điều này khiến các mô hình hồi quy logistic trở nên lý tưởng đối với những tổ chức đang bắt đầu với các dự án ML để đạt được một số thành tựu nhanh chóng.

**Sự linh hoạt**

Bạn có thể sử dụng hồi quy logistic để tìm đáp án cho các câu hỏi có hai hoặc nhiều kết quả hữu hạn. Bạn cũng có thể sử dụng phương pháp này để xử lý trước dữ liệu. Ví dụ: bạn có thể sắp xếp dữ liệu với một phạm vi giá trị lớn, chẳng hạn như giao dịch ngân hàng, thành một phạm vi giá trị hữu hạn, nhỏ hơn nhờ hồi quy logistic. Sau đó, bạn có thể xử lý tập dữ liệu nhỏ hơn này với các kỹ thuật ML khác để phân tích chính xác hơn.

**Khả năng hiển thị**

Phân tích hồi quy logistic cung cấp cho nhà phát triển khả năng nhìn nhận các quy trình phần mềm nội bộ rõ hơn so với các kỹ thuật phân tích dữ liệu khác. Khắc phục sự cố và sửa lỗi cũng trở nên dễ dàng hơn do các phép toán ít phức tạp hơn

### 2.5.2Nhược điểm

Giả định mối quan hệ tuyến tính giữa log-odds và các biến độc lập, dễ bị ảnh hưởng bởi các điểm dữ liệu ngoại lệ.

# CHƯƠNG 3: XỬ LÝ DỮ LIỆU VÀ TRAIN MODEL

## 3.1 Mô tả dataset

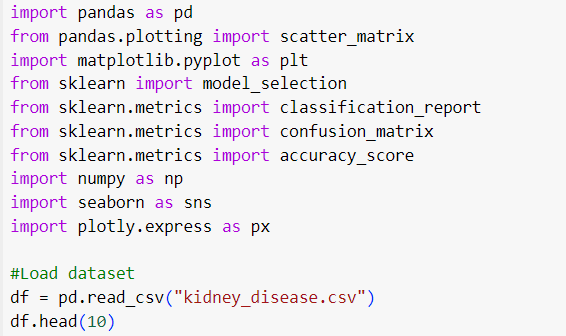
Bộ dữ liệu kidney\_desease.csv là bộ dữ liệu trả về kết quả một người có nguy cơ mắc bệnh thận không dựa vào các thuộc tính như độ tuổi, huyết áp, trọng lượng cơ thể, lượng đường,…gồm 25 cột và 400 dòng.

24 cột đầu là các thuộc tính có ảnh hưởng đến bệnh thận. Cột cuối cùng classfication là cột trả về kết quả một người có mắc bệnh thận hay không.

## 3.2 Các bước thực hiện

### 3.2.1 Tiền xử lý dữ liệu

**Bước 1:** Import thư viện cần thiết sklearn,… và load dữ liệu



**Bước 2: Khám phá tập dữ liệu:**

**df['classification'].value\_counts():**đếm số lần xuất hiện của từng giá trị trong cột 'classification' của DataFrame df.

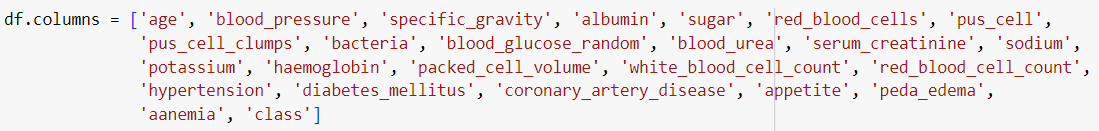


**df.drop('id',axis=1, inplace=True):** xóa cột 'id' khỏi DataFrame df. axis=1 chỉ định xóa theo cột, inplace=True  thay đổi sẽ được thực hiện trực tiếp trên DataFrame df, không tạo bản sao mới.



**df.head():** Hiển thị 5 dòng đầu tiên của DataFrame df.

**Bước 3: Đổi tên cột**



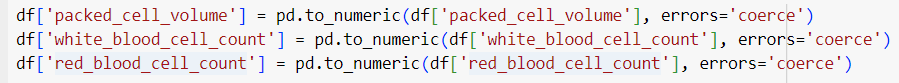
**df.head():** Hiển thị 5 dòng đầu tiên của DataFrame df với các cột đã được đổi tên.

**df.describe():** Hiển thị các thống kê mô tả (như min, max, mean, std, etc.) của các cột numeric trong DataFrame df.

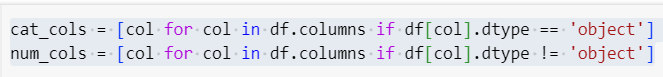
**df.info():** Hiển thị thông tin về DataFrame df, bao gồm số dòng, số cột, kiểu dữ liệu của các cột.

**Bước 4: Làm sạch và tiền xử lí dữ liệu**

1. **df['packed\_cell\_volume'] =** pd.to\_numeric(df['packed\_cell\_volume'], errors='coerce') và những câu lệnh tương tự: Các câu lệnh này sẽ chuyển đổi kiểu dữ liệu của các cột 'packed\_cell\_volume', 'white\_blood\_cell\_count', 'red\_blood\_cell\_count' từ dạng text sang dạng số. errors='coerce' sẽ chuyển các giá trị không thể chuyển đổi sang NaN (Not a Number).



**cat\_cols = [col for col in df.columns if df[col].dtype == 'object']**



tạo ra một danh sách mới được gọi là cat\_cols sau đó duyệt qua từng cột coltrong DataFrame df.

Với mỗi cột col, chúng tôi kiểm tra loại dữ liệu của cột đó bằng cách sử dụng df[col].dtype.

Nếu cột dữ liệu loại là object (tức là ký tự chuỗi), thì cột đó sẽ được thêm vào cat\_cols.

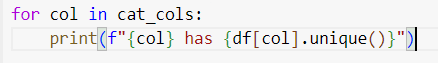
Cuối cùng, cat\_colssẽ chứa tên của tất cả các cột có 'đối tượng' dữ liệu loại trong DataFrame df.

num\_cols = [col for col in df.columns if df[col].dtype != 'object']

Tương tự, tạo ra một danh sách mới được gọi là num\_cols.

Lần này, họ kiểm tra xem loại dữ liệu của mỗi cột colcó object khác hay không.

Nếu cột dữ liệu kiểu không phải là object thì cột đó num\_cols.



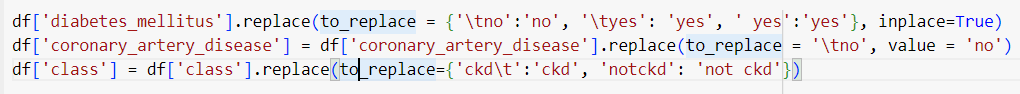
**for col in cat\_cols:** Chúng ta sử dụng vòng lặp for để duyệt qua từng cột col trong danh sách cat\_cols (các cột categorical) mà chúng ta đã tính toán ở trước đó.

**print(f"{col} has {df[col].unique()}"):** Với mỗi cột col, chúng ta thực hiện 2 việc:

df[col]: Truy cập vào cột col trong DataFrame df.

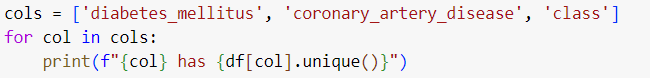
df[col].unique(): Lấy ra danh sách các giá trị duy nhất (unique values) của cột đó.

Sau đó, chúng ta in ra câu văn bằng cách sử dụng f-string, với nội dung "cột col có các giá trị duy nhất là df[col].unique()".



Chuẩn hóa các giá trị trong một số công cụ cột của DataFrame df, loại bỏ các ký tự không mong muốn như tab dấu và dấu cách.

1. df['diabetes\_mellitus'].replace(to\_replace = {'\tno':'no', '\tyes': 'yes', ' yes':'yes'}, inplace=True)
   * thao tác với cột "diabetes\_mellitus" trong DataFrame df.
   * sử dụng phương pháp replace() để thay thế một số công cụ có giá trị:
     + \tno(không có tab dấu) được thay thế bằng 'no'.
     + \tyes(có với tab dấu) được thay thế bằng 'yes'.
     + ' yes'(có dấu cách phía sau) được thay thế bằng 'yes'.
   * inplace=Truecó nghĩa là sự thay đổi sẽ được thực hiện trực tiếp trên DataFrame df, không cần tạo bản sao mới.
2. df['coronary\_artery\_disease'] = df['coronary\_artery\_disease'].replace(to\_replace = '\tno', value = 'no')
   * Ở đây, chúng tôi đang thao tác với cột "coronary\_artery\_disease" trong DataFrame df.
   * Chúng tôi sử dụng phương pháp replace()để thay thế giá trị '\tno'(không có dấu tab) bằng 'no'.
   * Kết quả được phép thay thế được gán lại vào cột "coronary\_artery\_disease" trong DataFrame df.
3. df['class'] = df['class'].replace(to\_replace={'ckd\t':'ckd', 'notckd': 'not ckd'})
   * Ở đây, chúng tôi đang thao tác với cột "class" trong DataFrame df.
   * Chúng tôi sử dụng phương thức replace()để thay thế hai giá trị:
     + 'ckd\t'(ckd with tab dấu) được thay thế bằng 'ckd'.
     + 'notckd'được thay thế bằng 'not ckd'.
   * Kết quả được phép thay thế được gán lại vào "class" cột trong DataFrame df.

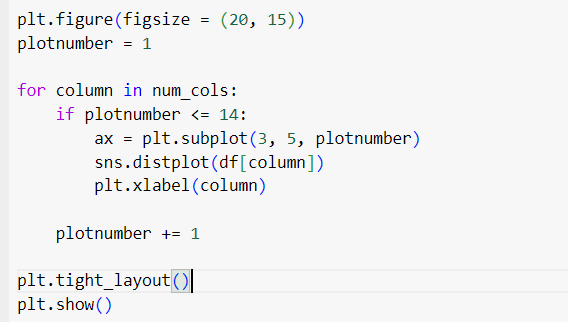


1. **Xác định các cột cần kiểm tra**: Bằng cách tạo một danh sách cols chứa các tên cột mà chúng ta muốn kiểm tra, như **['diabetes\_mellitus', 'coronary\_artery\_disease', 'class'],** chúng ta có thể dễ dàng thay đổi danh sách các cột cần quan tâm mà không phải sửa đổi toàn bộ đoạn code.
2. **In ra các giá trị duy nhất (unique values) của các cột**: Với mỗi cột trong danh sách cols, chúng ta sử dụng **df[col].unique()** để lấy ra danh sách các giá trị duy nhất trong cột đó. Sau đó, chúng ta in ra thông tin này bằng cách sử dụng f-string, tạo ra một output dễ đọc như "cột col có các giá trị duy nhất là df[col].unique()".



1. **Mã hóa biến phụ thuộc (target variable) 'class'**:
   * Dòng df['class'] = df['class'].map({'ckd':0, 'not ckd': 1}) thực hiện việc mã hóa các giá trị chuỗi (string) trong cột 'class' thành các giá trị số học (0 và 1).
   * Mục đích của việc mã hóa này là để chuẩn bị dữ liệu cho các thuật toán machine learning, vì hầu hết các mô hình đều yêu cầu biến phụ thuộc phải là giá trị số.
   * Trong trường hợp này, giá trị '0' được gán cho class 'ckd' (chronic kidney disease) và giá trị '1' được gán cho class 'not ckd'.
2. **Chuyển đổi cột 'class' thành kiểu dữ liệu số**:
   * Dòng df['class'] = pd.to\_numeric(df['class'], errors = 'coerce') chuyển đổi cột 'class' thành kiểu dữ liệu số (numeric).
   * Tham số errors='coerce' được sử dụng để xử lý các giá trị không thể chuyển đổi thành số. Trong trường hợp này, các giá trị không thể chuyển đổi sẽ được đổi thành giá trị NaN (Not a Number).
   * Việc chuyển đổi cột 'class' thành kiểu dữ liệu số là cần thiết vì hầu hết các thuật toán machine learning yêu cầu biến phụ thuộc phải là kiểu số.

**Bước 5: Phân tích dữ liệu thăm dò**



1. **Tạo figure với kích thước cụ thể**:

* plt.figure(figsize = (20, 15)) tạo ra một figure (canvas) có kích thước 20 inches chiều rộng và 15 inches chiều cao. Điều này giúp tạo ra một bố cục rộng rãi và dễ nhìn cho các subplot sẽ được vẽ.

1. **Khởi tạo biến theo dõi số subplot**:

* plotnumber = 1 khởi tạo biến plotnumber để theo dõi thứ tự của các subplot sẽ được vẽ.

1. **Vẽ các biểu đồ phân phối (distribution plots)**:

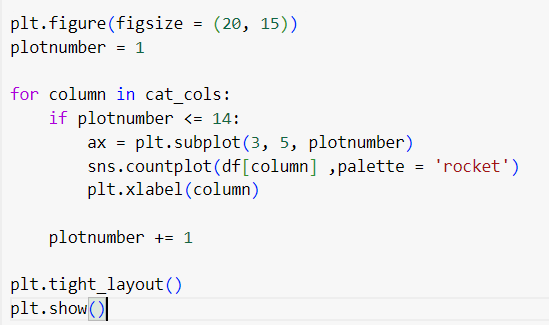
* Sử dụng vòng lặp for column in num\_cols: để lặp qua các cột (numeric features) trong danh sách num\_cols.
* Với mỗi cột, nếu plotnumber nhỏ hơn hoặc bằng 14 (tức là chỉ vẽ 14 subplot đầu tiên):
* Tạo một subplot mới bằng plt.subplot(3, 5, plotnumber), với 3 hàng, 5 cột và vị trí subplot là plotnumber.
* Vẽ biểu đồ phân phối (distribution plot) cho cột đó bằng sns.distplot(df[column]).
* Thêm nhãn (label) cho trục x là tên của cột bằng plt.xlabel(column).
* Tăng giá trị của plotnumber lên 1 để chuẩn bị cho subplot tiếp theo.

1. **Sắp xếp lại bố cục subplot**:

* plt.tight\_layout() sắp xếp lại kích thước và vị trí của các subplot để chúng vừa khít và không bị tràn ra khỏi figure.

1. **Hiển thị figure**:

* plt.show() hiển thị toàn bộ figure đã được vẽ.



1. **Tạo figure với kích thước cụ thể**:
   * plt.figure(figsize = (20, 15)) tạo ra một figure (canvas) có kích thước 20 inches chiều rộng và 15 inches chiều cao.
2. **Khởi tạo biến theo dõi số subplot**:
   * plotnumber = 1 khởi tạo biến plotnumber để theo dõi thứ tự của các subplot sẽ được vẽ.
3. **Vẽ các biểu đồ đếm (count plots)**:
   * Sử dụng vòng lặp for column in cat\_cols: để lặp qua các cột (categorical features) trong danh sách cat\_cols.
   * Với mỗi cột, nếu plotnumber nhỏ hơn hoặc bằng 14 (tức là chỉ vẽ 14 subplot đầu tiên):
     + Tạo một subplot mới bằng plt.subplot(3, 5, plotnumber), với 3 hàng, 5 cột và vị trí subplot là plotnumber.
     + Vẽ biểu đồ đếm (count plot) cho cột đó bằng sns.countplot(df[column], palette='rocket'). Sử dụng palette 'rocket' để tạo màu sắc cho biểu đồ.
     + Thêm nhãn (label) cho trục x là tên của cột bằng plt.xlabel(column).
   * Tăng giá trị của plotnumber lên 1 để chuẩn bị cho subplot tiếp theo.
4. **Sắp xếp lại bố cục subplot**:
   * plt.tight\_layout() sắp xếp lại kích thước và vị trí của các subplot để chúng vừa khít và không bị tràn ra khỏi figure.
5. **Hiển thị figure**:
   * plt.show() hiển thị toàn bộ figure đã được vẽ.

**Bước 6: Xử lí các giá trị bị thiếu**

kiểm tra mục tiêu và hiển thị thiếu giá trị số lượng

1. **df.isnull()** tạo ra một DataFrame mới có cùng kích thước như df, nhưng chứa các giá trị boolean bị thiếu, trong đó True biểu thị cho giá trị bị thiếu và False biểu thị cho giá trị không bị thiếu.
2. .sum() tính tổng số giá trị True
3. .sort\_values(ascending=False)sắp xếp chuỗi này theo thứ tự giảm dần (thứ tự giảm dần) về việc thiếu giá trị số lượng,





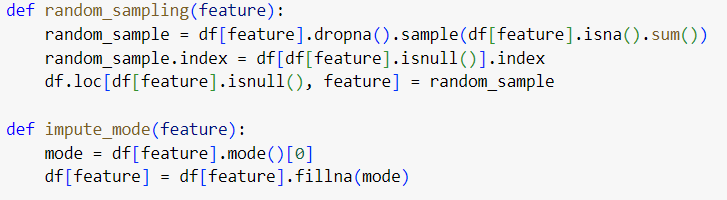


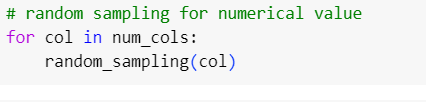
1. **random\_sampling(feature):**
   1. Mục đích: Thay thế các giá trị thiếu trong cột feature của DataFrame df bằng các giá trị ngẫu nhiên lấy từ những giá trị không thiếu trong cùng cột đó.
   2. Cách thức:

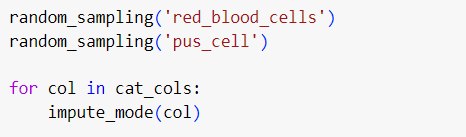
* Lấy ra một mẫu ngẫu nhiên (sample()) từ những giá trị không thiếu (dropna()) trong cột feature, với số lượng bằng số lượng giá trị thiếu (df[feature].isna().sum()).
* Gán lại index của mẫu ngẫu nhiên này bằng index của các giá trị thiếu trong cột feature (df[df[feature].isnull()].index).
* Thay thế các giá trị thiếu trong cột feature bằng các giá trị trong mẫu ngẫu nhiên vừa tạo.

1. **impute\_mode(feature):**
   1. Mục đích: Thay thế các giá trị thiếu trong cột feature của DataFrame df bằng giá trị mode (giá trị xuất hiện nhiều nhất) của cột đó.
   2. Cách thực hiện:

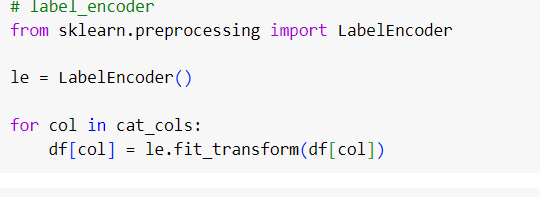
* Tính giá trị mode của cột feature (df[feature].mode()[0]).
* Thay thế các giá trị thiếu trong cột feature bằng giá trị mode vừa tính (df[feature] = df[feature].fillna(mode)).





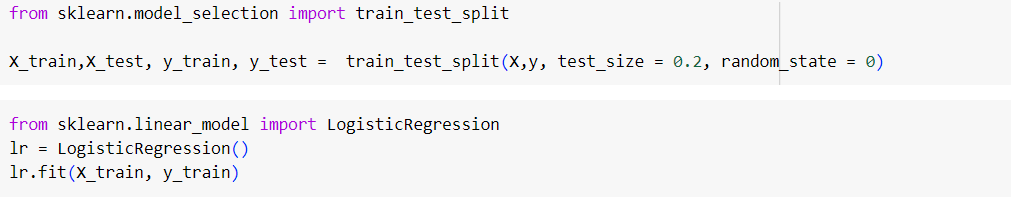


**Bước 7: Mã hóa nhãn**

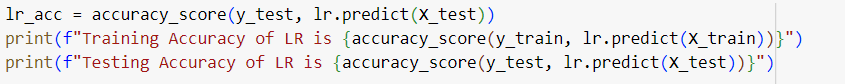


* le.fit\_transform(df[col]): Mã hóa các giá trị phân loại trong cột col thành các giá trị số (0, 1, 2, ...) dựa trên sự xuất hiện của các giá trị duy nhất.
* df[col] = ...: Ghi lại kết quả mã hóa vào cùng cột col trong DataFrame df.

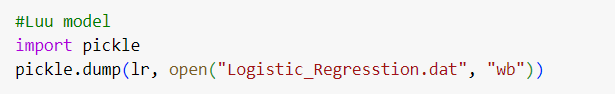
### 3.2.2 Train model Logictic Regresstion



1. Huấn luyện mô hình học máy trên tập huấn luyện ( X\_train, y\_train).
2. Đánh giá hiệu suất của mô hình trên tập thử nghiệm chưa nhìn thấy ( X\_test, y\_test).
3. Chia bộ dữ liệu thành 20% test và 80% train
4. Cuối cùng in ra Accuracy của train và test



**Bước 9: Lưu mô hình**

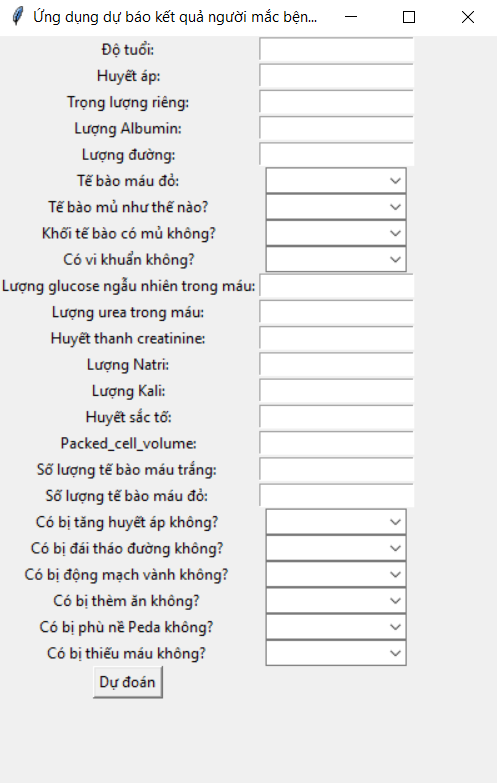


# CHƯƠNG 4: XÂY DỰNG ỨNG DỤNG

## 4.1 Mô tả ứng dụng

Ứng dụng dự báo một người có mắc bệnh thận không khi người dùng(các bác sỹ, nhân viên y tế) truyền vào các tham số độ tuổi, huyết áp, trọng lượng riêng, lượng Albumin, lượng đường, số lượng tế bào máu trắng, số lượng tế bào máu đỏ, có thèm ăn hay không, có phù nề hay không, có bị tăng huyết áp hay không,... Ứng dụng sẽ đưa ra dự báo kết quả người đó có mắc bệnh thận mãn tính không. Từ đó giúp các bác sỹ đưa ra những quyết định đúng đắn để đưa ra các phương án điều trị cho bệnh nhân.

## 4.2 Giao diện ứng dụng



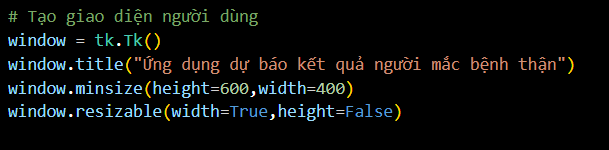
Giao diện được viết bằng thư viện Tkinker. Tkinker là một thư viện chuẩn của Python được sử dụng rộng rãi cho phát triển ứng dụng GUI.

Để tạo được giao diện người dùng ta làm như sau:

Bước 1: Import thư viện Tkinker vào Python

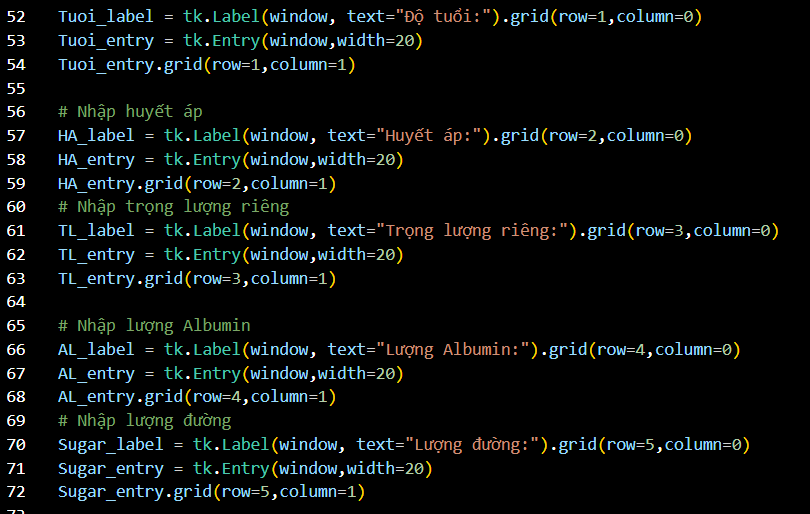


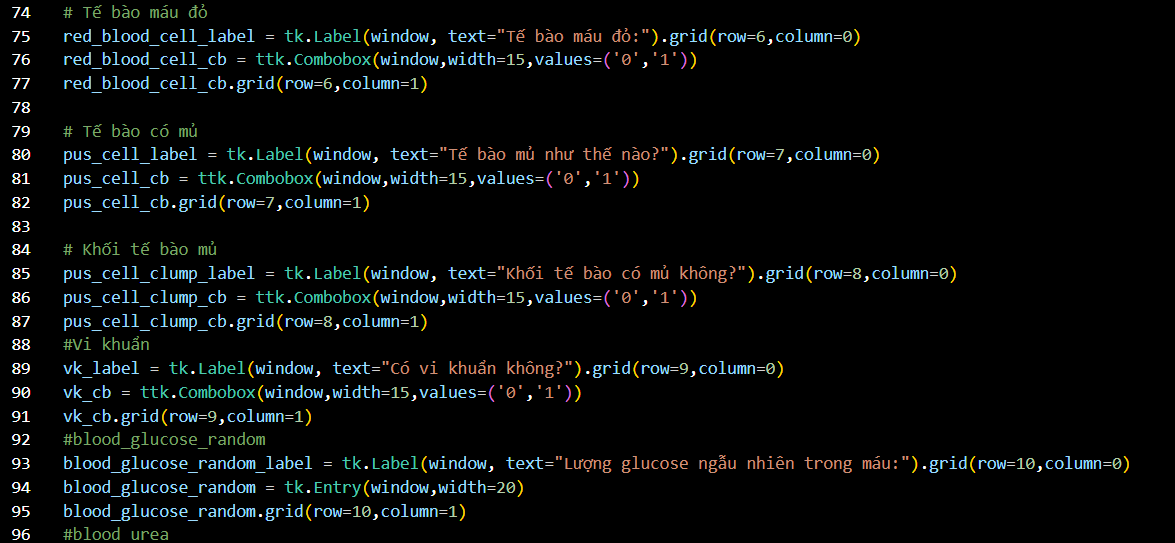
Bước 2: Sau đó ta tạo một cửa sổ gốc:



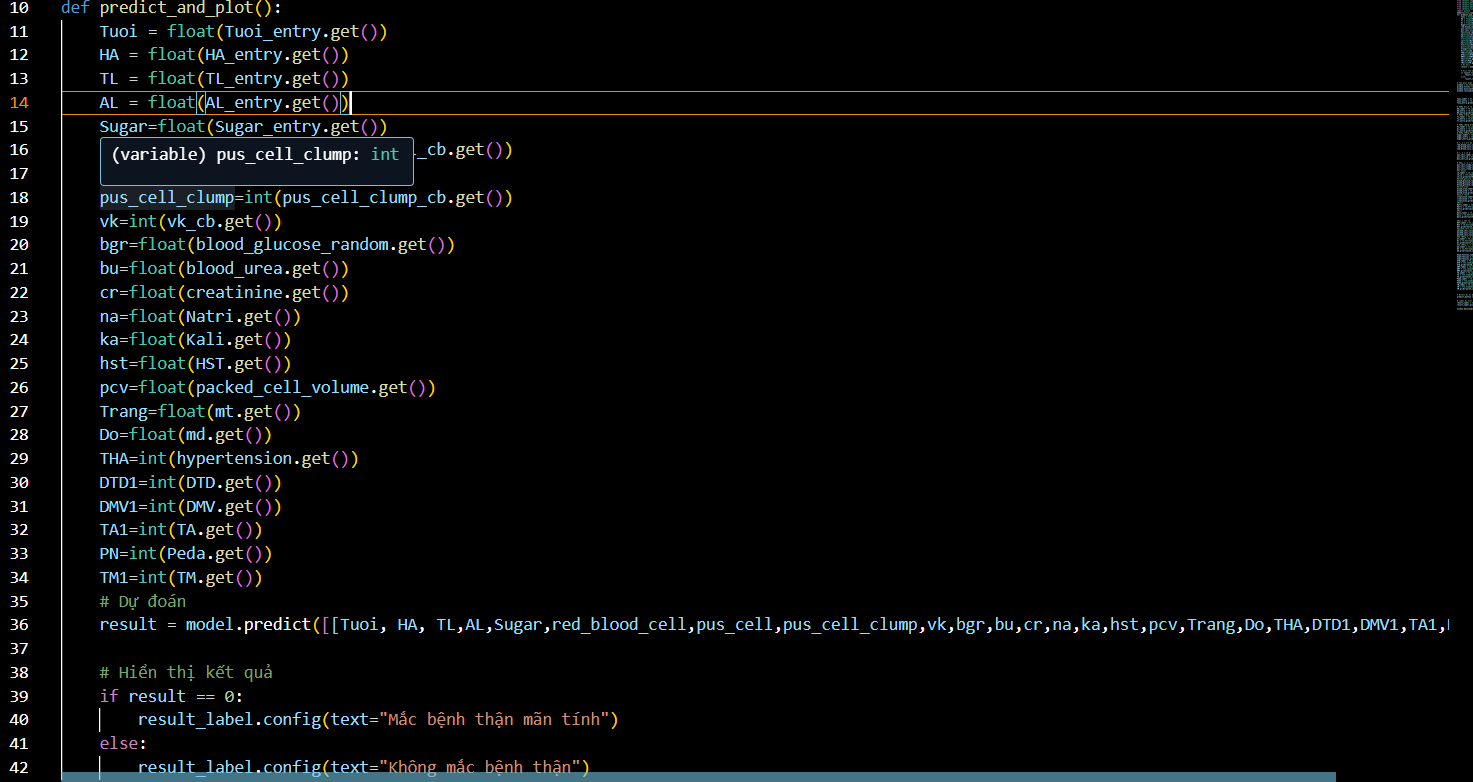
Bước 3: Tạo Form

Ở bước này tạo form cho ứng dụng gồm các label, textbox và button.





Bước 4: Xử lí hàm predict\_and\_plot để xuất ra kết quả dự đoán người mắc bệnh và vẽ biểu đồ



Ta sẽ khai báo các biến Tuoi, HA(Huyết áp), TL(Trọng lượng), AL(lượng Albumin), Sugar(lượng đường),…để lấy các giá trị người dùng nhập ở ô textbox. Sau đó khai báo một biến là kết quả gọi lại mô hình đã train ở chương 3 để dự đoán và cho xuất kết quả ra.

# TÀI LIỆU THAM KHẢO

[Bài 6: Logistic Regression (Hồi quy Logistic) - Trí tuệ nhân tạo (trituenhantao.io)](https://trituenhantao.io/machine-learning-co-ban/bai-6-logistic-regression-hoi-quy-logistic/)