TRƯỜNG ĐẠI HỌC MỞ HÀ NỘI

**KHOA CÔNG NGHỆ THÔNG TIN**

----------------------



**BÁO CÁO BÀI TẬP LỚN**

**MÔN: NHẬP MÔN KHAI PHÁ DỮ LIỆU VÀ MÁY HỌC**

**ĐỀ TÀI: PHÂN LOẠI NẤM ĐỘC DỰA TRÊN PHƯƠNG PHÁP HỌC MÁY CÓ GIÁM SÁT**

***Giảng viên hướng dẫn***: ***TS Vũ Xuân Hạnh***

***Nhóm sinh viên thực hiện***:

Nguyễn Tuấn Vũ – 2110A02

Mai Thị Thảo Chi – 2110A03

Đàm Phương Dung – 2110A03

Đỗ Ngọc Giao – 2110A03

**Hà Nội – 2024**

**PHÂN CÔNG CÔNG VIỆC VÀ KẾ HOẠCH THỰC HIỆN**

| **Họ và tên** | **Công việc** | **Thời gian thực hiện** |
| --- | --- | --- |
| Nguyễn Tuấn Vũ | Tổng quan về đề tài, Làm sạch dữ liệu  (Tiền xử lý dữ liệu), Tích hợp mô hình học máy vào ứng dụng | 15/03/2024 - 1/04/2024 |
| Đàm Phương Dung | Tổng quan về đề tài, Xây dựng mô hình học máy (Decision Tree), Thu thập dữ liệu (Tiền xử lý dữ liệu), Hoàn thiện báo cáo. | 15/03/2024 - 12/04/2024 |
| Mai Thị Thảo Chi | Tổng quan về đề tài, (Tiền xử lý dữ liệu), Đánh giá mô hình học máy | 15/03/2024 - 12/04/2024 |
| Đỗ Ngọc Giao | Tổng quan về đề tài, Tích hợp và biến đổi dữ liệu (Tiền xử lý dữ liệu), Xây dựng mô hình học máy (Cây quyết định), Đánh giá mô hình học máy | 15/03/2024 - 12/04/2024 |

**MỤC LỤC**

[**I. TỔNG QUAN VỀ ĐỀ TÀI 1**](#_gjdgxs)

[1. Lý do chọn đề tài 1](#_30j0zll)

[2. Tổng quan về Học máy 1](#_1fob9te)

[3. Chu trình Học máy 2](#_3znysh7)

[4. Học máy có giám sát 3](#_2et92p0)

[**II. TIỀN XỬ LÝ DỮ LIỆU 3**](#_tyjcwt)

[1. Xác định vấn đề 3](#_3dy6vkm)

[2. Xác định mục đích 3](#_1t3h5sf)

[3. Đặc điểm dữ liệu 3](#_4d34og8)

[4. Phương pháp thu thập dữ liệu 4](#_2s8eyo1)

[5. Tiền xử lý dữ liệu 4](#_17dp8vu)

[5.1. Sửa lại lỗi font dữ liệu tải về 4](#_3rdcrjn)

[5.2. Làm sạch dữ liệu 5](#_3rdcrjn)

[5.3. Tích hợp và chuyển đổi dữ liệu 10](#_26in1rg)

[**III. XÂY DỰNG MÔ HÌNH HỌC MÁY 16**](#_35nkun2)

[1. Decision Tree (Cây quyết định) 16](#_3j2qqm3)

[1.1. Cơ sở lý thuyết 16](#_1y810tw)

[1.1.1. Decision trong bài toán Classification 17](#_8ddf3evz7nwl)

[1.1.2. Một số thông số quan trọng của Decision Tree 18](#_x0b4k5dnmr5v)

[1.1.3. Cách Decision Tree hoạt động 19](#_2uczpu45555l)

[1.2. Vì sao chọn Decision Tree cho bài toán 20](#_4i7ojhp)

[1.3. Áp dụng cơ sở lý thuyết vào xây dựng mô hình cho bài toán 20](#_2xcytpi)

[2. Random forest 20](#_1ksv4uv)

[2.1. Cơ sở lý thuyết 20](#_44sinio)

[2.2. Vì sao chọn Random forest cho bài toán 21](#_2jxsxqh)

[2.3. Áp dụng cơ sở lý thuyết vào xây dựng mô hình cho bài toán 21](#_z337ya)

[**IV. ĐÁNH GIÁ MÔ HÌNH HỌC MÁY 22**](#_3whwml4)

[1. Kiểm tra chéo đánh giá hiệu suất 2 mô hình (thực hiện kiểm tra chéo 10 lần) 22](#_3wsfpwg6ljsw)

[2. Đánh giá khi chạy mô hình test và mô hình train 24](#_y6f46nk30rd4)

[2.1 Decision tree 24](#_pi5nxiavp3dq)

[2.2 Random Forest 25](#_7l5ah6vtj5v3)

[3. Đánh giá mô hình thông qua ma trận nhầm lẫn (Confusing Matrix). 26](#_o3zc3mh6klit)

[**V. TÍCH HỢP MÔ HÌNH HỌC MÁY VÀO ỨNG DỤNG 29**](#_2bn6wsx)

[**VI. KẾT LUẬN 34**](#_qsh70q)

# 

# TỔNG QUAN VỀ ĐỀ TÀI

## Lý do chọn đề tài

Học máy là môn khoa học nhằm phát triển những thuật toán và mô hình thống kê mà các hệ thống máy tính sử dụng để thực hiện các tác vụ dựa vào khuôn mẫu và suy luận mà không cần hướng dẫn cụ thể. Các hệ thống máy tính sử dụng thuật toán máy học để xử lý khối lượng lớn dữ liệu trong quá khứ và xác định các khuôn mẫu dữ liệu. Việc này cho phép chúng dự đoán kết quả chính xác hơn từ cùng một tập dữ liệu đầu vào cho trước.

Trong lĩnh vực an toàn thực phẩm, các mô hình học máy để dự đoán các vấn đề thực phẩm chung và dự đoán độc hại nói riêng luôn nhận được sự quan tâm của cộng đồng nghiên cứu, và luôn dành được sự ưu tiên trong việc đóng góp cảnh báo nguy cơ độc hại của thực phẩm, cụ thể ở đề tài chúng em chọn là NẤM.

Ứng dụng thực tế: Kỹ thuật phân loại nấm độc có thể được áp dụng rộng rãi trong lĩnh vực y tế, sinh thái học, và nông nghiệp để phát hiện và phân biệt nấm độc và nấm ăn được.

Đóng góp cho cộng đồng khoa học: Nghiên cứu và phát triển trong lĩnh vực phân loại nấm độc có thể đóng góp vào hiểu biết về loài nấm, cũng như cung cấp công cụ hữu ích cho các nhà khoa học và cộng đồng.

Từ những lý do ở trên, chúng em quyết định lựa chọn đề tài ***“Phân loại nấm độc dựa trên phương pháp học máy có giám sát”***

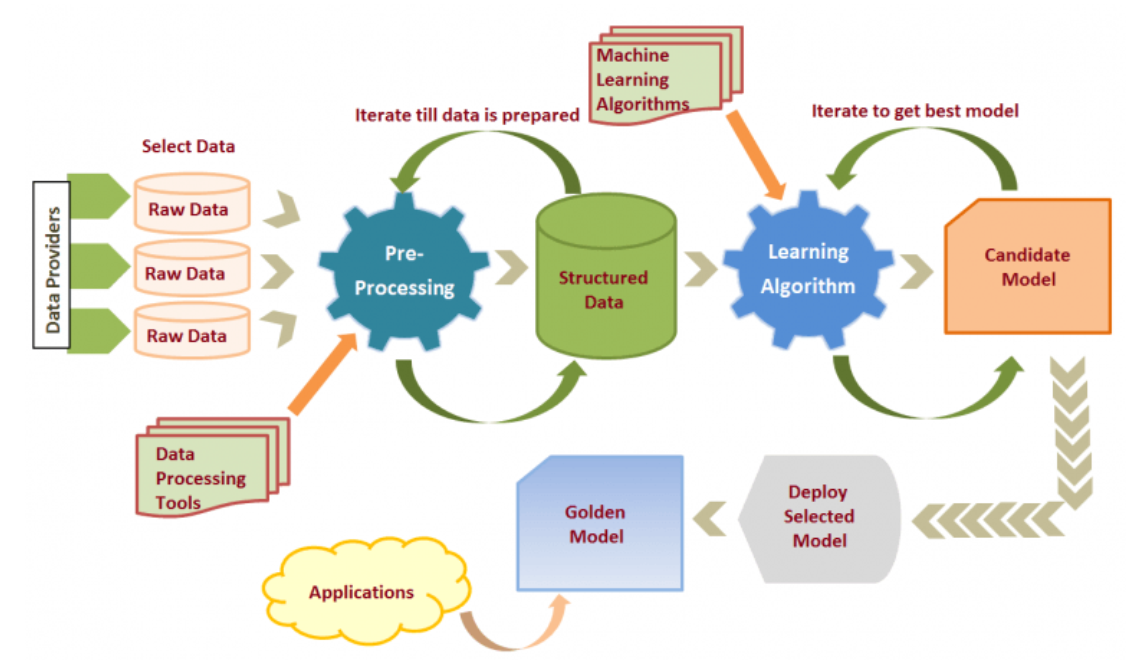
## Tổng quan về Học máy

Học máy (machine learning) là một lĩnh vực của trí tuệ nhân tạo liên quan đến việc nghiên cứu và xây dựng các kĩ thuật cho phép các hệ thống “học” tự động từ dữ liệu để giải quyết những vấn đề cụ thể.

Một cách tổng quát, trong cuốn sách Machine Learning của tác giả Tom Mitchell xuất bản năm 1997, học máy được định nghĩa như sau: “A computer program is said to learn to perform a task T from experience E, if its performance at task T, as measured by a performance metric P, improves with experience E over time” (một chương trình máy tính được cho là học để thực hiện một nhiệm vụ T từ kinh nghiệm E, nếu hiệu suất thực hiện công việc T của nó được đo bởi chỉ số hiệu suất P và được cải thiện bởi kinh nghiệm E theo thời gian).

Như vậy, học máy nói đến một chương trình giúp cho máy tính có thể giải một bài toán cụ thể bằng những “kinh nghiệm” mà nó đã được “học” từ dữ liệu.

## Chu trình Học máy



* **Giai đoạn 1:** Tiền xử lý dữ liệu

Giai đoạn này, tập trung ở việc thu thập dữ liệu, sau đó tiến hành làm sạch dữ liệu(điền các giá trị còn thiếu, xác định các ngoại lệ như dữ liệu bị trùng lặp, dữ liệu không nhất quán và làm mịn dữ liệu nhiễu), tích hợp và chuyển đổi dữ liệu(tích hợp nhiều dataset, chuẩn hóa và tổng hợp dữ liệu), giảm dữ liệu(giảm bớt về các thuộc tính của dữ liệu, giảm bớt kích thước dữ liệu – nhưng vẫn đảm bảo thu được các kết quả khai phá dữ liệu tương đương) và rời rạc hóa.

* **Giai đoạn 2:** Xử lý dữ liệu

Trong quá trình này, chúng ta cũng cần phải thực hiện phân chia dữ liệu thành các tập khác nhau bao gồm: 1 tập huấn luyện (training set), tập tối ưu (validation set) và tập kiểm thử (test set) với mục đích đo đạc, chọn ra mô hình với thông số phù hợp và kiểm tra độ chính xác của mô hình trong thực tế.

* **Giai đoạn 3:** Xây dựng mô hình và điều chỉnh tham số

Mục đích của bước này là xây dựng các mô hình cho việc huấn luyện. Các tham số của các mô hình sẽ được điều chỉnh phù hợp dựa trên kết quả học của tập huấn luyện và tập tối ưu.

* **Giai đoạn 4:** Kiểm thử trên tập thử nghiệm

Sau đó, các mô hình tốt nhất đại diện cho mỗi thuật toán sẽ được đưa vào đánh giá hiệu quả trên tập kiểm thử, sử dụng các phân tích, đánh giá để lựa chọn mô hình tốt nhất (Golden Model).

* **Giai đoạn 5:** Áp dụng mô hình học máy vào ứng dung

Cuối cùng, sau khi chọn được mô hình, chúng ta sẽ áp dụng mô hình học máy vào ứng dụng để người sử dụng có thể nhập các thông tin.

## Học máy có giám sát

Học có giám sát là trường hợp dữ liệu huấn luyện được cho một cách tường minh dưới dạng đầu vào và đầu ra của hàm đích, ví dụ, cho trước tập các mẫu cùng nhãn phân loại tương ứng.

Hàm y = f(x) từ tập huấn luyện cho trước {x1, x2,…, xn; y1, y2,…, yn} sao cho yi ≅ f(xi) với mọi i

Học máy khi đó được gọi là có giám sát để thể hiện việc thuật toán nhận được chỉ dẫn trực tiếp về lời giải cho từng trường hợp.

Việc học máy có giám sát có thể được nhóm lại thành các vấn đề về phân loại và hồi quy.

* Phân loại (Classification): Việc phân loại diễn ra khi biến đầu ra là một thể loại nào đó, chẳng hạn như “đỏ” hoặc “xanh” hoặc “bệnh” và “không có bệnh”.
* Hồi quy (Regression): Việc hồi quy xảy ra là khi biến đầu ra là một giá trị thực, chẳng hạn như “đô la” hay “trọng lượng”.

# TIỀN XỬ LÝ DỮ LIỆU

## Xác định vấn đề

* Nhận biết được đâu là nấm độc
* Giúp con người có kiến thức an toàn về nấm

## Xác định mục đích

* Xác định các yếu tố ảnh hưởng
  + - Các đặc điểm, hình thái, thuộc tính của nấm
* Xác định mục đích:
  + - Phát triển một mô hình dự đoán có thể xác định với độ chính xác cao khoảng trên 90% khả năng nấm là nấm độc dựa trên các đặc điểm thuộc tính của nấm.
    - Giảm tỷ lệ con người không biết nấm có độc xuống dưới 10%

## Đặc điểm dữ liệu

* Các thông tin mà dữ liệu cần có:
  + - Các đặc điểm thuộc tính của nấm
    - Kết quả sau phân tích
      * Yes (nấm độc)
      * No (nấm ăn được)
* Để xây dựng mô hình đạt được hiệu quả tốt nhất, hoàn thành các mục đích đề ra, khối lượng của dữ liệu phải khoảng từ 5000 – 10000 các bản ghi dữ liệu
* Độ chính xác và tin cậy chính xác, được đảm bởi các nguồn uy tín như Kaggle, thông tin của chính phủ, bộ y tế cung cấp

## Phương pháp thu thập dữ liệu

Phương pháp thu thập dữ liệu là từ nguồn hiện có, cụ thể được ký hiệu trong báo cáo như sau

[DATASET 1] Mushroom Classification, thu thập dữ liệu từ Kaggle <https://www.kaggle.com/datasets/uciml/mushroom-classification>

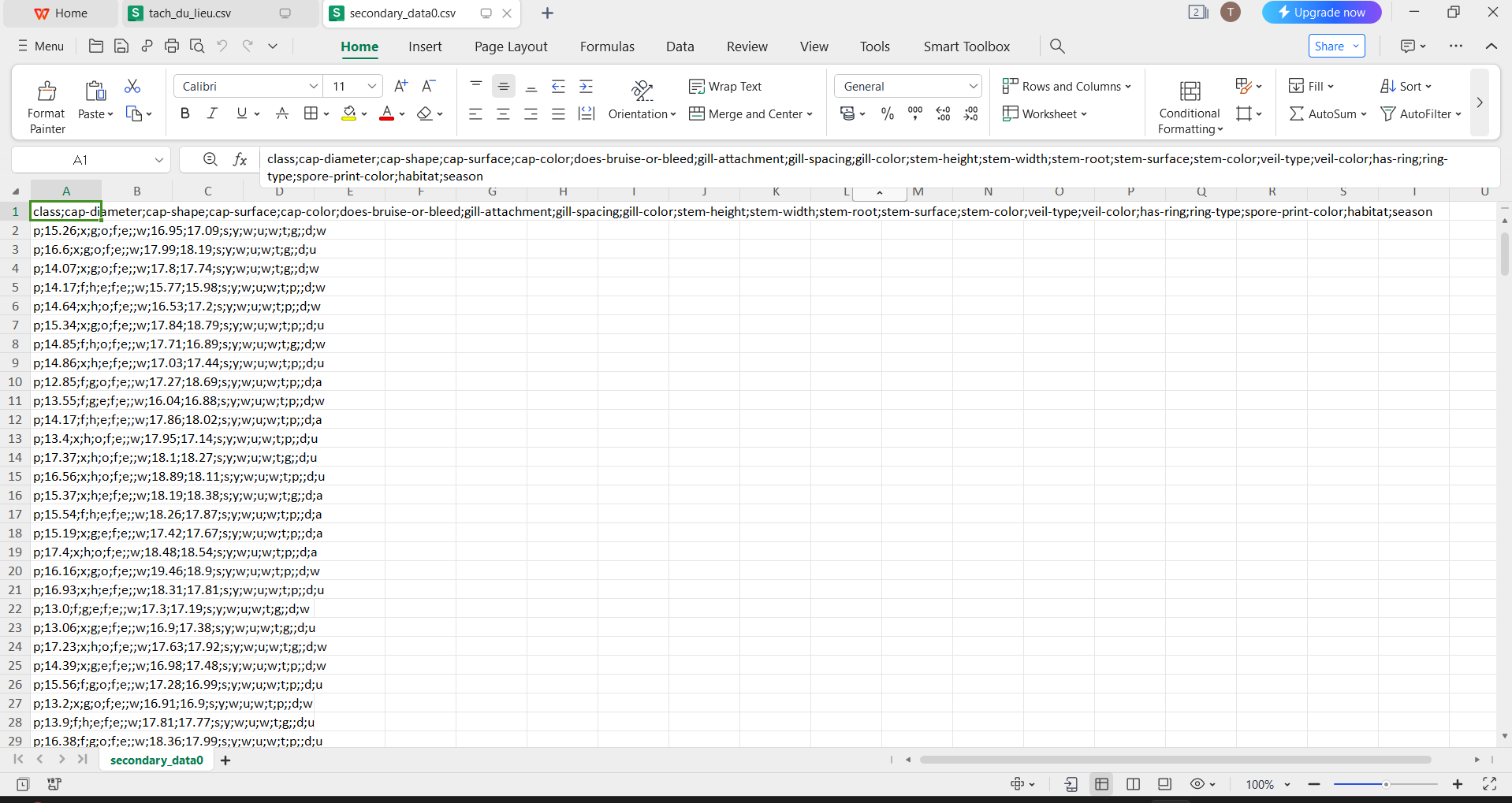
[DATASET 2] Mushroom Edibility Classification từ Kaggle <https://www.kaggle.com/datasets/devzohaib/mushroom-edibility-classification>

## Tiền xử lý dữ liệu

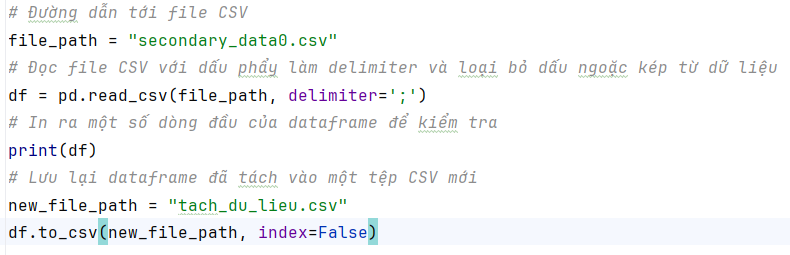
### Sửa lại lỗi font dữ liệu tải về

File dữ liệu csv tải về bị lỗi font dữ liệu

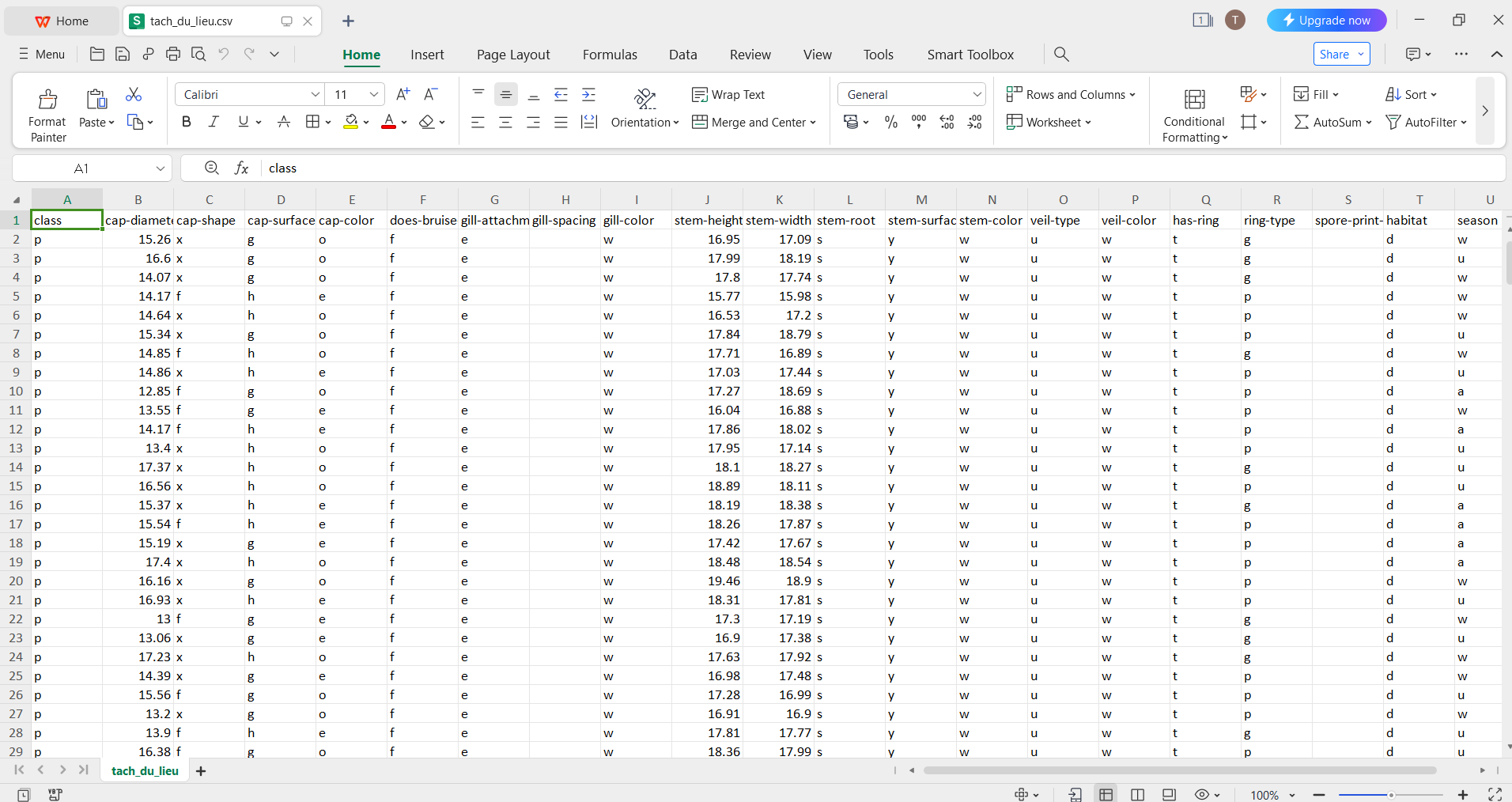
* DATASET 1: không bị
* DATASET 2: Bị lỗi trên



Áp dụng code sau:

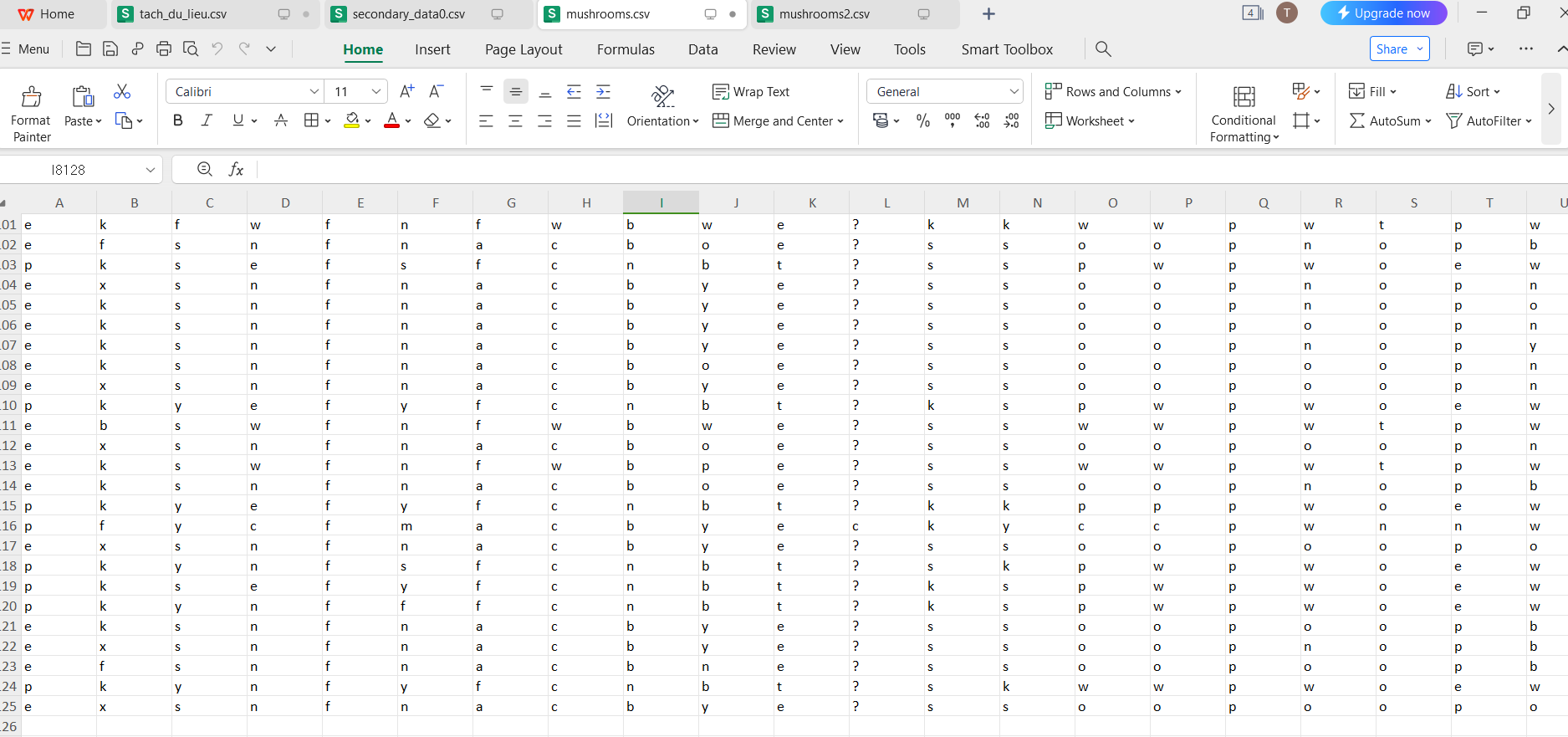
****

Kết quả:

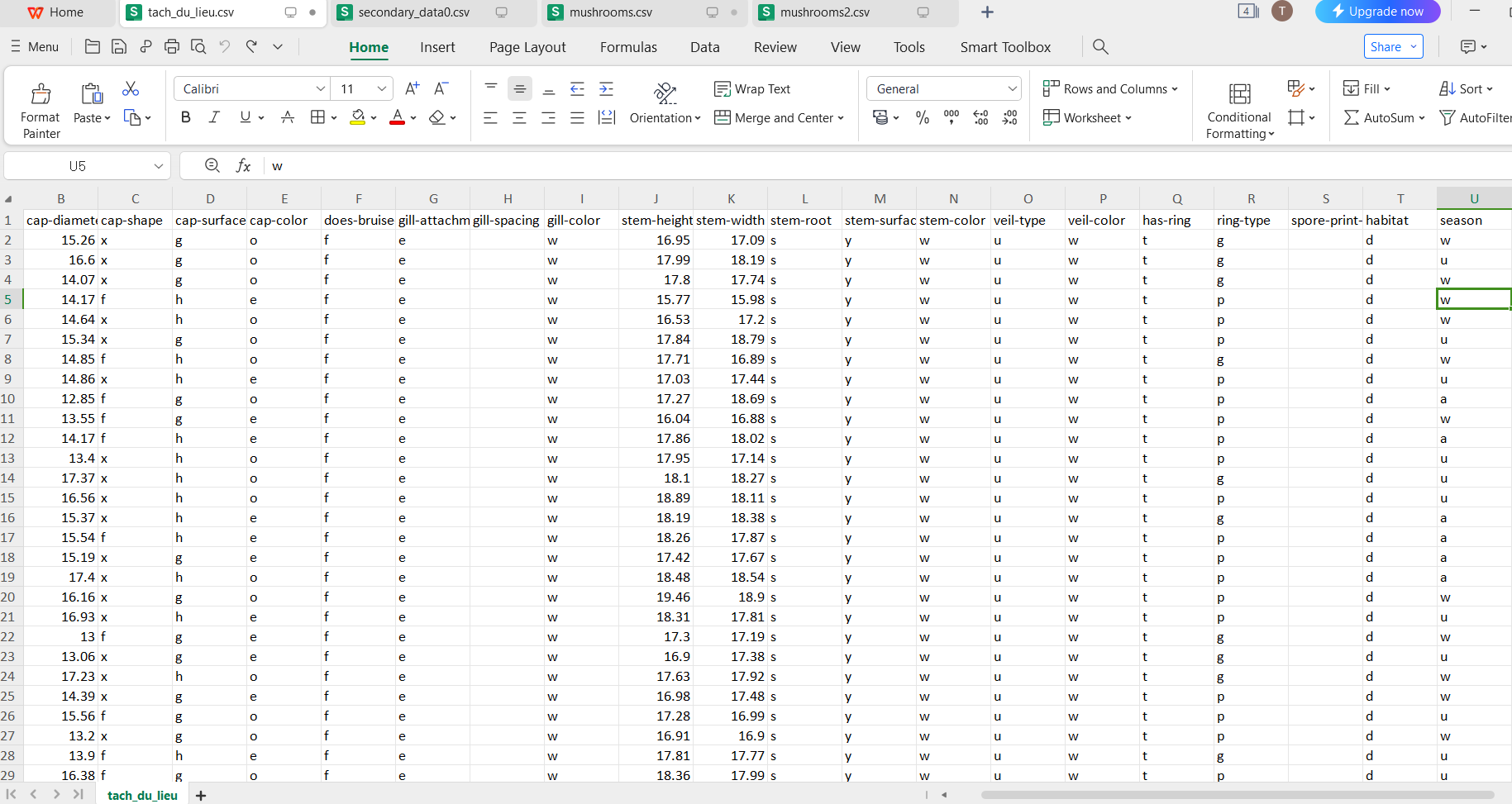


### Làm sạch dữ liệu

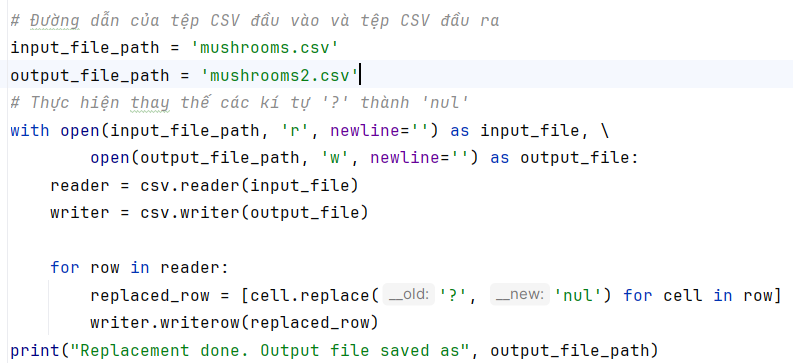
* Điền vào các giá trị còn thiếu, thay thế kí tự “?”
  + - Phương pháp: thay giá trị còn thiếu = nul, “?” = nul
    - Cách thực hiện
      * Bước 1: Xác định giá trị còn thiếu
* Với DATASET 1, có kí tự “?” : 2480/ 186852

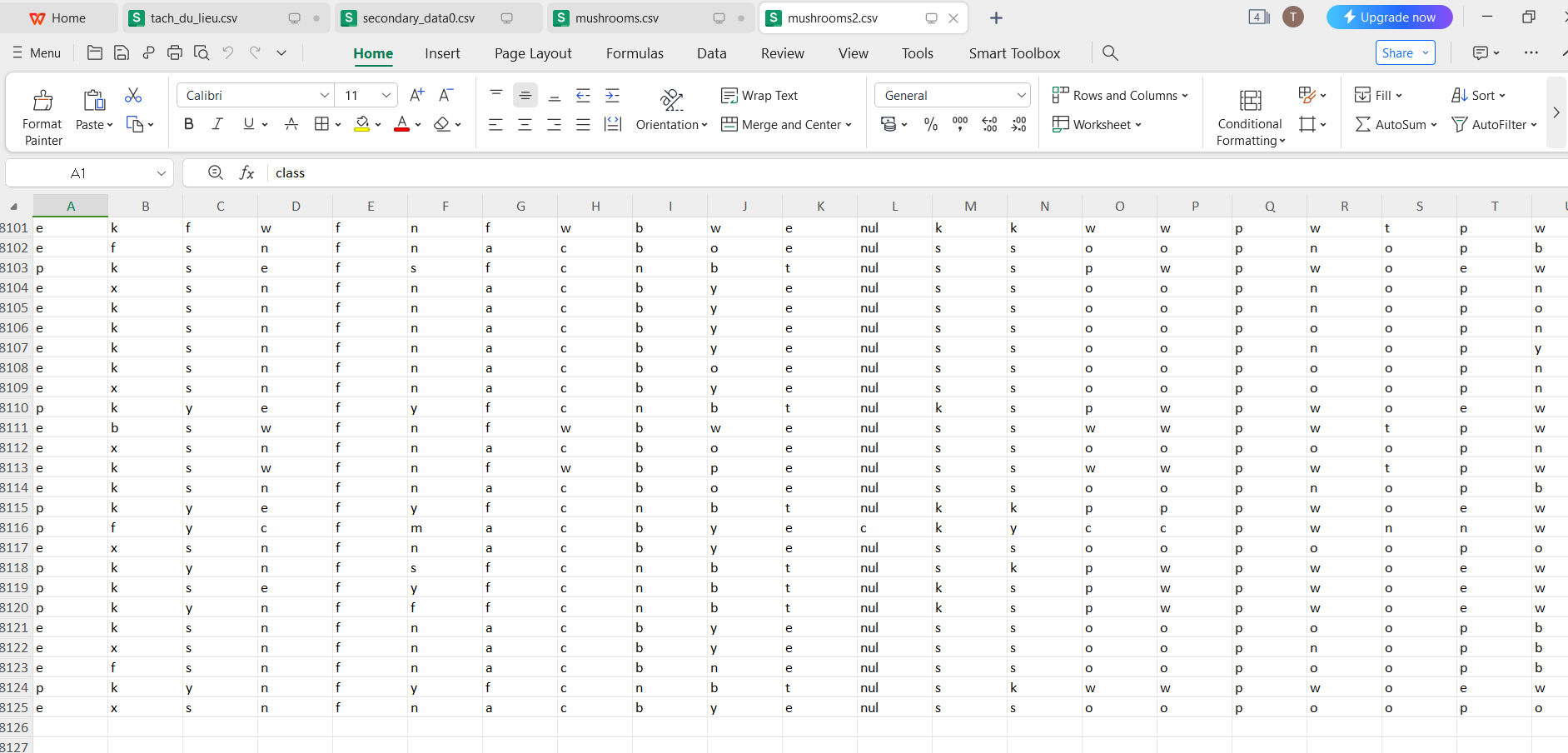


* Với DATASET 2, có giá trị thiếu 307463/ 1282449 ô trống

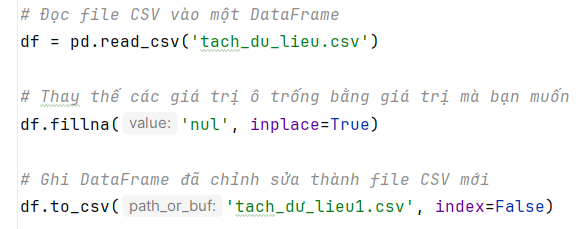


* Bước 2: Thực hiện thay thế
  + Code sử dụng (DATASET 1)

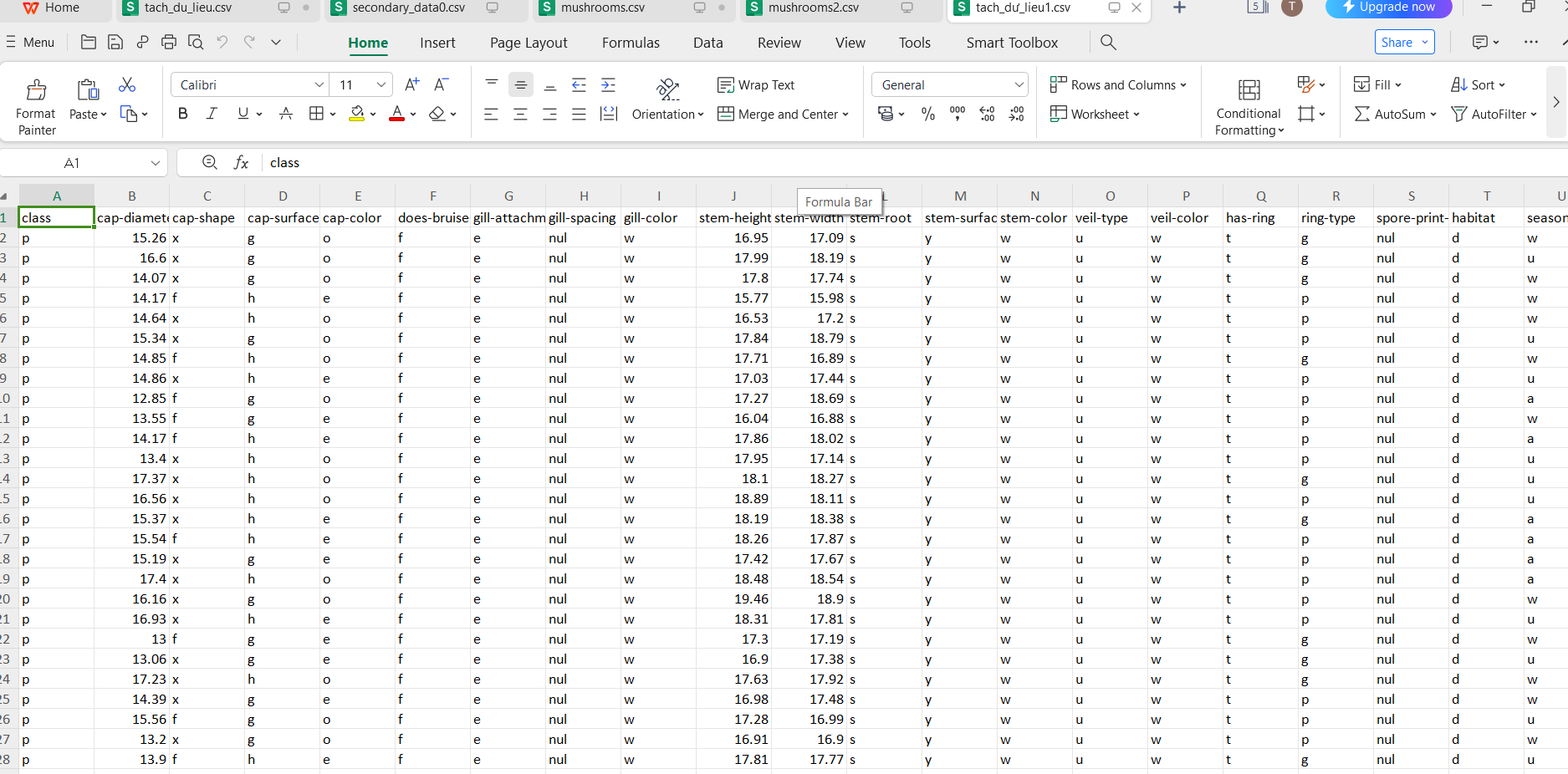


Kết quả:

* + Code sử dụng (DATASET 2)

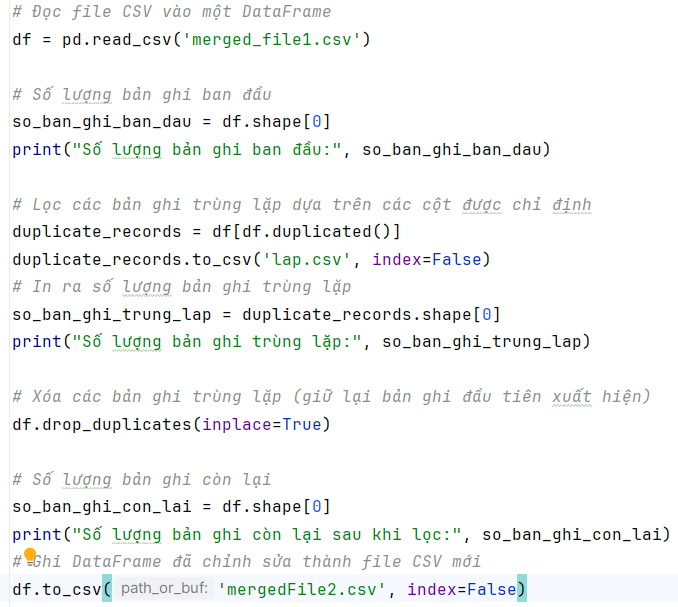


Kết quả:

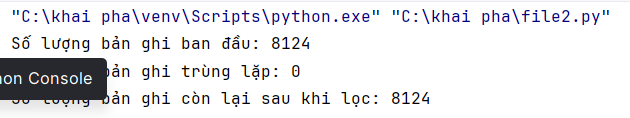


* Lọc các bản ghi trùng lặp
  + - Sử dụng hàm drop\_duplicates() để xác định các bản ghi trùng lặp và loại bỏ các bản ghi đó
    - Hàm drop\_duplicates() sử dụng một thuật toán băm (hashing) để xác định các bản ghi có cùng giá trị cho tất cả các cột được chỉ định. Sau khi đã xác định được các bản ghi trùng lặp thì tiến hành loại bỏ chúng khỏi dataset.

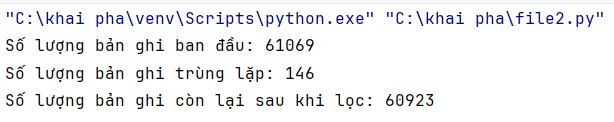
* + - Code lọc các bản ghi trùng lặp sử dụng hàm drop\_duplicates()



* + - Kết quả
      * Với DATASET 1:



* + - * Với DATASET 2:

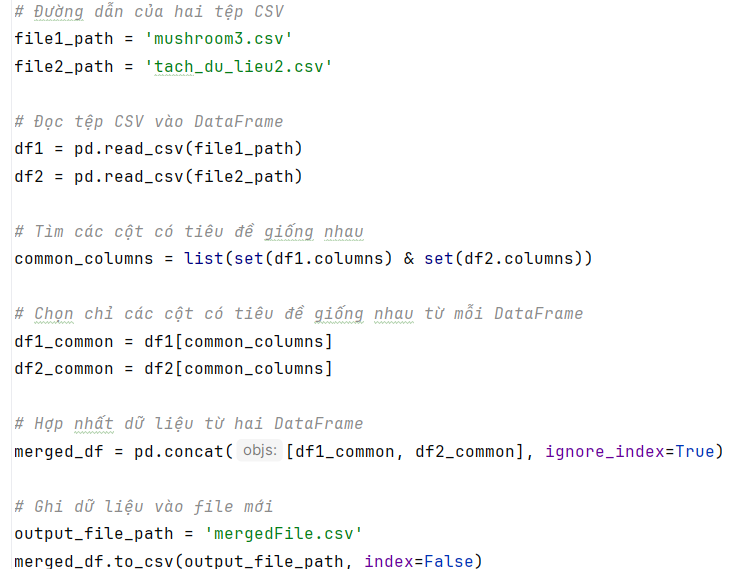


### Tích hợp và chuyển đổi dữ liệu

* Data Integration hay quá trình tích hợp dữ liệu bao gồm việc kết hợp các dữ liệu không đồng nhất trong những nguồn khác nhau vào trong cùng một lược đồ duy nhất.
* Quá trình tích hợp dữ liệu
  + - Xác định các trường dữ liệu giống nhau giữa 2 DATASET

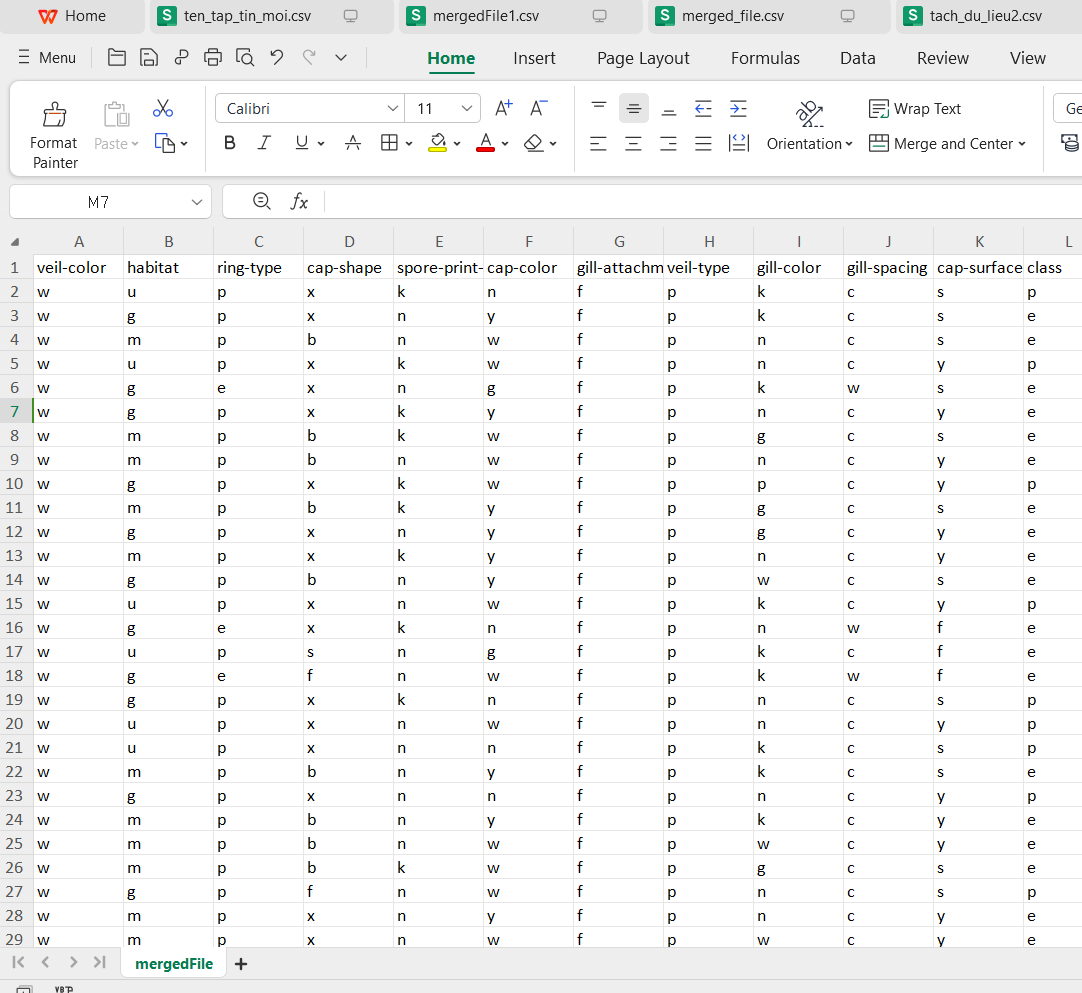
| **DATASET 1** | **DATASET 2** |
| --- | --- |
| cap-shape:bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s | cap-diameter: float number in cm |
| cap-surface:fibrous=f, grooves=g, scaly=y,smooth=s | cap-shape: bell=b, conical=c, convex=x, flat=f, sunken=s, spherical=p, others=o |
| cap-color: brown=n ,buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u,red=e, white=w, yellow=y | cap-surface: fibrous=i, grooves=g, scaly=y, smooth=s, shiny=h, leathery=l, silky=k, sticky=t,  wrinkled=w, fleshy=e |
| bruises: bruises=t,no=f | cap-color (n): brown=n, buff=b, gray=g, green=r, pink=p,  purple=u, red=e, white=w, yellow=y, blue=l,  orange=o, black=k |
| odor: almond=a, anise=l,creosote=c, fishy=y,foul=f,musty=m,none=n,pungent=p,spicy=s | does-bruise-bleed (n): bruises-or-bleeding=t,no=f |
| gill-attachment:attached=a, descending=d,free=f,notched=n | gill-attachment (n): adnate=a, adnexed=x, decurrent=d, free=e,  sinuate=s, pores=p, none=f, unknown= nul |
| gill-spacing:close=c,crowded=w,distant=d | gill-spacing (n):close=c, distant=d, none=f |
| gill-size: broad=b,narrow=n | gill-color (n):see cap-color + none=f |
| gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g,green=r,orange=o,pink=p,purple=u,red=e,white=w,yellow=y | stem-height (m):float number in cm |
| stalk-shape: enlarging=e,tapering=t | stem-width (m):float number in mm |
| stalk-root: bulbous=b,club=c,cup=u,equal=e,rhizomorphs=z,rooted=r,missing=nul | stem-root (n):bulbous=b, swollen=s, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r |
| stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s | stem-surface (n):see cap-surface + none=f |
| stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s | stem-color (n):see cap-color + none=f |
| stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y | veil-type (n):partial=p, universal=u |
| stalk-color-below-ring: brown=n, buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y | veil-color (n):see cap-color + none=f |
| veil-type: partial=p,universal=u | has-ring (n):ring=t, none=f |
| veil-color:brown=n, orange=o, white=w ,yellow=y | ring-type (n):cobwebby=c, evanescent=e, flaring=r, grooved=g,  large=l, pendant=p, sheathing=s, zone=z, scaly=y, movable=m, none=f, unknown= nul |
| ring-number: none=n,one=o,two=t | spore-print-color (n): see cap color |
| ring-type:cobwebby=c,evanescent=e, flaring=f,large=l,none=n,pendant=p,sheathing=s,zone=z | habitat (n):grasses=g, leaves=l, meadows=m, paths=p, heaths=h, urban=u, waste=w, woods=d |
| spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r,orange=o,purple=u,white=w,yellow=y | season: spring=s, summer=u, autumn=a, winter=w |
| population: abundant=a,clustered=c,numerous=n,scattered=s,several=v,solitary=y | class: edible=e, poisonous=p |
| habitat: grasses=g,leaves=l,meadows=m,paths=p,urban=u,waste=w,woods=d |  |
| class: edible=e, poisonous=p |  |

* + - Tích hợp dữ liệu
      * Xóa các cột không trùng nhau ở cả 2 bảng bằng delete column trong Excel
      * Tích hợp 2 DATASET sử dụng hàm concat()

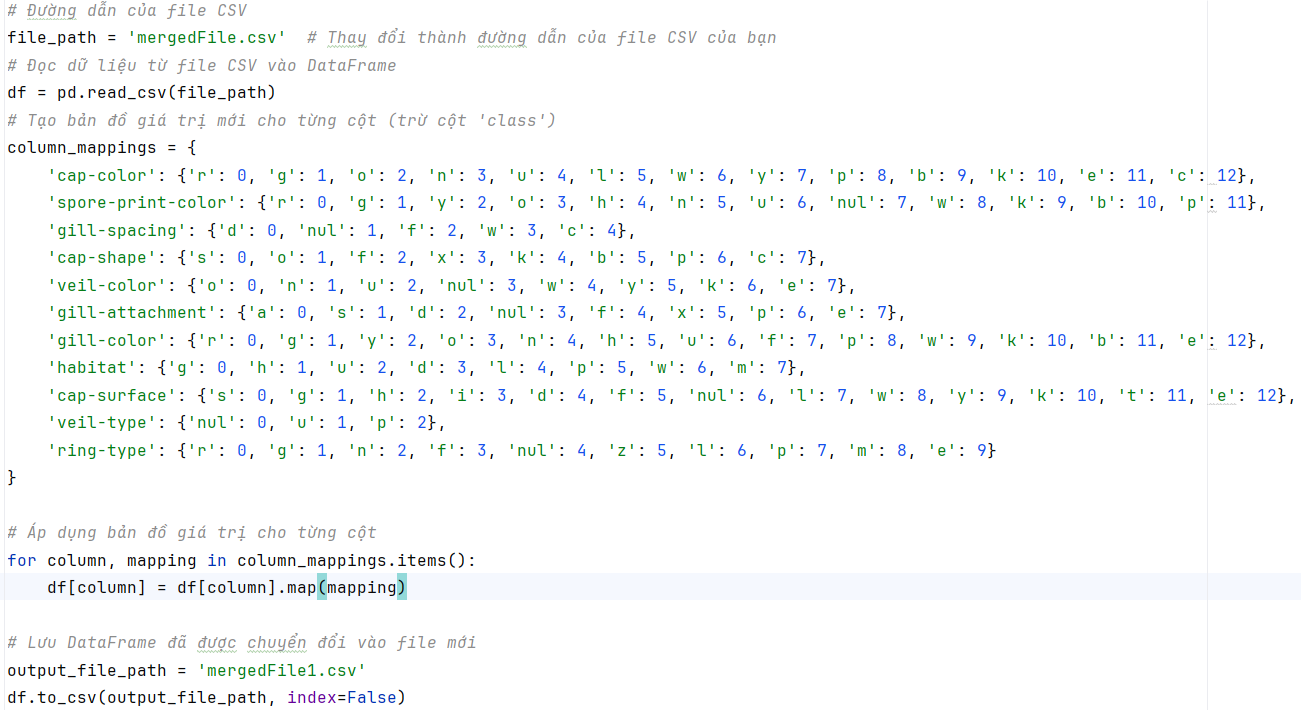


* + - * Kết quả DATASET mới sau khi tích hợp

DATASET mới bao gồm 69047 bản ghi, 11 thuộc tính và 1 nhãn.

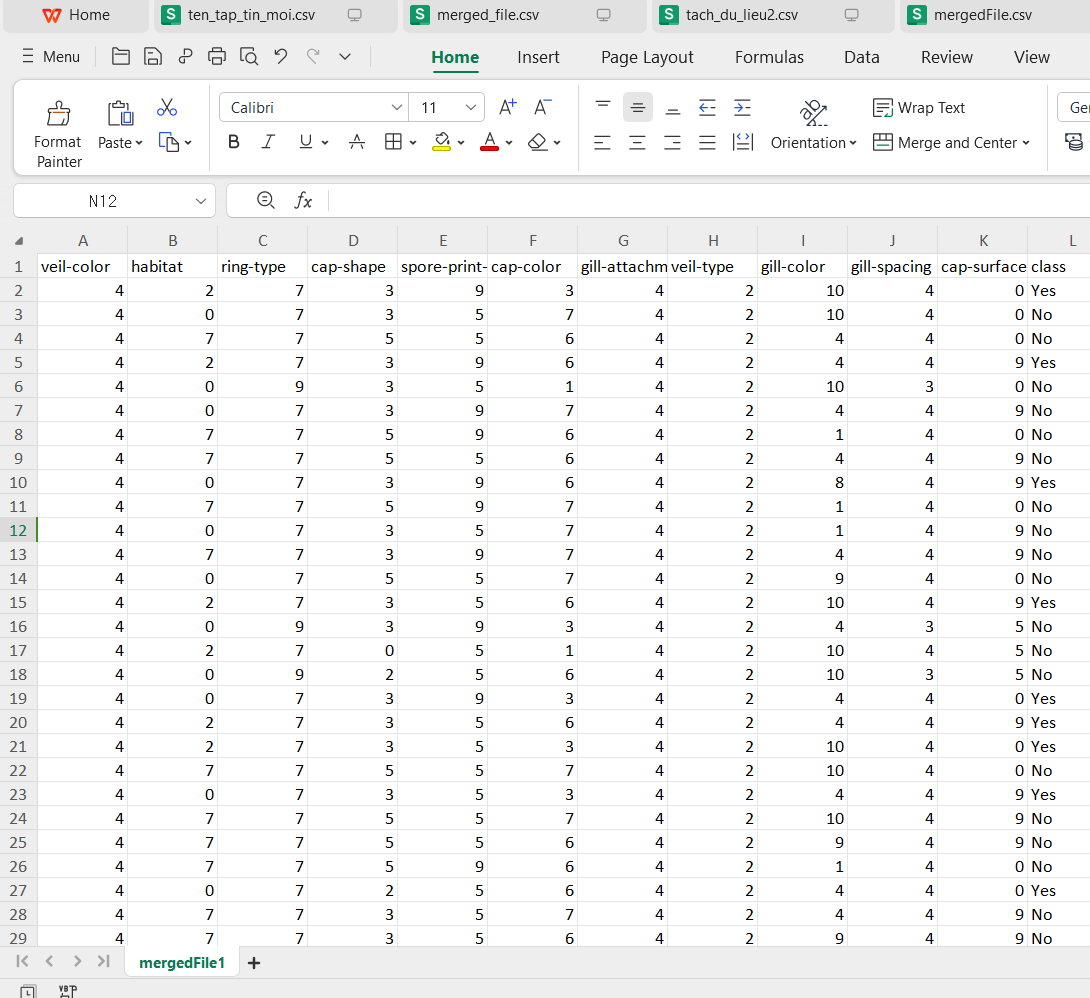


* Chuyển dữ liệu sang dạng số: dùng bản đồ mapping để gán giá trị dữ liệu sang dạng số, và dùng map() để thực hiện việc chuyển đổi
  + Code thực hiện:



Ta gán p=Yes, e=No trong cột class bằng Replace trong Excel

Kết quả:



# XÂY DỰNG MÔ HÌNH HỌC MÁY

## Decision Tree (Cây quyết định)

### Cơ sở lý thuyết

* Mô hình cây quyết định là một mô hình được sử dụng khá phổ biến và hiệu quả trong cả hai lớp bài toán Classification và Regression của học có giám sát. Khác với những thuật toán khác trong học có giám sát, mô hình cây quyết định không tồn tại phương trình dự báo. Mọi việc chúng ta cần thực hiện đó là tìm ra một cây quyết định dự báo tốt trên tập train và sử dụng cây quyết định này dự báo trên tập kiểm tra.
* Mỗi nhánh của cây giống như nhánh if, else nên có thể hiểu cây quyết định là tập hợp các luật nếu thì
* Cây quyết định nhận đầu vào là một bộ giá trị thuộc tính mô tả một đối tượng hay một tình huống nào đó và trả về một giá trị rời rạc.
* Mỗi bộ thuộc tính đầu vào được gọi là một mẫu hay một ví dụ, đầu ra gọi là loại hay nhãn phân loại. Thuộc tính đầu vào còn được gọi là đặc trưng và có thể nhận giá trị rời rạc hoặc liên tục.
* Để cho đơn giản, trước tiên sẽ xem xét thuộc tính rời rạc, sau đó sẽ mở rộng cho trường hợp thuộc tính nhận giá trị liên tục.
* Thuật toán:
  + - Input: tập dữ liệu huấn luyện
    - Output: Cây quyết định
      * Khởi đầu: nút hiện thời là nút gốc chứa toàn bộ tập dữ liệu huấn luyện
      * Tại nút hiện thời n, lựa chọn thuộc tính:

Chưa được sử dụng (tức là nút nằm trên đường đi từ gốc tới nút hiện thời)

Cho phép phân chia tập dữ liệu hiện thời thành các tập con tốt nhất

* + - * Với mỗi giá trị thuộc tính được chọn:

Thêm một nút con bên dưới

Chia mẫu ở nút hiện thời về các nút con theo giá trị thuộc tính được chọn

* + - * Lặp (đệ quy) với mỗi nút con cho đến khi:

Tất cả các thuộc tính đã được sử dụng ở các nút phía trên, hoặc

Tất cả ví dụ tại nút hiện thời có cùng nhãn phân loại nhãn của nút được lấy theo đa số nhãn của ví dụ tại nút hiện thời

#### Decision trong bài toán Classification

* Biểu diễn dữ liệu:
  + - Mỗi đối tượng quan sát được thể hiện bởi một vector n chiều, mỗi chiều đại diện cho một thuộc tính đối tượng.
    - Mỗi đỉnh bên trong cây biểu diễn một thuộc tính để kiểm tra dữ liệu đến về sau
    - Mỗi nhánh xuất phát từ một đỉnh tương ứng với mỗi một giá trị trong miền giá trị của thuộc tính đó
    - Mỗi lá lưu nhãn lớp
    - Cho dữ liệu cần phán đoán duyệt qua cây đấy, dùng các thuộc tính của nó duyệt từ gốc đến lá, lấy nhãn ở lá để phán đoán cho dữ liệu mới

#### Một số thông số quan trọng của Decision Tree

* Thuật toán xây dựng cây ID3 sử dụng entropy làm mức đo độ đồng nhất của tập dữ liệu. Trên cơ sở entropy, thuật toán tính độ tăng thông tin như mức tăng độ đồng nhất, từ đây xác định thuộc tính tốt nhất tại mỗi nút.
* Trong trường hợp chỉ có hai nhãn phân loại, ký hiệu là + và -, entropy H(S) của tập dữ liệu S được tính như sau:

H(S) = - p+log2p+ - p–log2p-

Trong đó p+ và p- là xác suất quan sát thấy nhãn phân loại + và -, tính bằng tần suất quan sát thấy + và – trong dữ liệu.

* Trong bảng huấn luyện trên, ta thấy có 9 nhãn “Yes” và 5 nhãn “No”, do đó:

+

* Có thể nhận thấy, trong trường hợp nhãn nhị phân, entropy đạt giá trị tối đa bằng 1 khi xác suất hai nhãn bằng nhau và bằng 0.5, entropy đạt giá trị nhỏ nhất bằng 0 khi xác suất một nhãn là 1 và nhãn còn lại là 0. Nói chung, entropy càng gần 0 thì tập đối tượng càng đồng nhất.
* Trong TH tổng quát với C nhãn phân loại có xác suất lần lượt là p1, p2, …, pc entropy được tính như sau:
* Giá trị cực đại của entropy khi đó sẽ bằng log2C khi các nhãn có xác suất như nhau và giá trị nhỏ nhất của entropy bằng 0 khi tất cả đối tượng có chung một nhãn. Tương tự trường hợp nhãn nhị phân, giá trị entropy càng nhỏ thì tập dữ liệu càng đồng nhất.
* Độ tăng thông tin (Information Gain), ký hiệu IG, là chỉ số đánh giá độ tốt của thuộc tính trong việc phân chia tập dữ liệu thành những tập con đồng nhất. IG được tính dựa trên entropy theo công thức sau:

Trong đó:

* + - S là tập dữ liệu ở nút hiện tại
    - A là thuộc tính values (A) là tập các giá trị của thuộc tính A
    - Sv là tập các mẫu có giá trị thuộc tính A bằng v
    - |S| và |Sv| là lực lượng của các tập hợp tương ứng

#### Cách Decision Tree hoạt động

Mô hình Decision Tree (cây quyết định) hoạt động bằng cách xây dựng một cây quyết định từ dữ liệu huấn luyện để dự đoán nhãn hoặc giá trị mục tiêu cho dữ liệu mới. Dưới đây là cách Decision Tree hoạt động chi tiết:

* Chọn biến phân chia: Quá trình bắt đầu từ nút gốc của cây, thuật toán chọn một biến đặc trưng và một giá trị phân chia sao cho phân chia dữ liệu thành các nhóm có độ thuần khiết tốt nhất có thể. Độ thuần khiết được đo lường bằng các phép đo như Gini impurity hoặc entropy.
* Tạo nút mới trên cây: Sau khi biến phân chia được chọn, một nút mới được tạo trên cây và dữ liệu được phân loại vào các nhánh con tương ứng dựa trên giá trị của biến phân chia.
* Lặp lại quá trình cho các nhánh con: Quá trình này được lặp lại đệ quy cho mỗi nhánh con, tạo ra các nút mới và phân chia dữ liệu trong từng nhánh con đến khi một điều kiện dừng được đạt đến. Điều kiện dừng có thể là đạt đến một độ sâu tối đa cho cây, đạt đến một độ thuần khiết nhất định, hoặc các điều kiện khác.
* Dừng xây dựng cây: Quá trình xây dựng cây kết thúc khi một điều kiện dừng được đạt đến. Tại các nút lá cuối cùng của cây, dữ liệu được phân loại vào các nhãn hoặc dự đoán giá trị cuối cùng
* Dự đoán cho dữ liệu mới: Khi cây đã được xây dựng, nó có thể được sử dụng để dự đoán nhãn hoặc giá trị cho các mẫu dữ liệu mới. Mẫu dữ liệu mới được đưa vào cây và đi theo đường dẫn từ nút gốc đến nút lá, và nhãn hoặc giá trị cuối cùng của nút lá đó sẽ là dự đoán cho mẫu dữ liệu mới.

### Vì sao chọn Decision Tree cho bài toán

Cây quyết định tạo ra một mô hình dễ hiểu, có thể diễn giải được bằng cách sử dụng các quy tắc và câu lệnh điều kiện đơn giản. Mô hình cây quyết định thường dễ kiểm tra với dữ liệu mới, và thậm chí còn cho phép kiểm tra tại thời điểm xây dựng cây.

### Áp dụng cơ sở lý thuyết vào xây dựng mô hình cho bài toán

* Đọc dữ liệu từ file CSV

data = pd.read\_csv('D:\kpdl\merged\_file1.csv')

* Phân chia dữ liệu thành 2 phần độc lập và phụ thuộc

X = data.loc[:, ['veil-color','habitat', 'ring-type', 'cap-shape', 'spore-print-color', 'cap-color', 'gill-attachment', 'veil-type','gill-color', "gill-spacing", "cap-surface"]]

y = data.loc[:, 'class']

* Chuyển đổi biến đầu vào thành giá trị nguyên

encoder = LabelEncoder()

for i in X.columns:

X[i] = encoder.fit\_transform(X[i])

y = encoder.fit\_transform(y)

* Xây dựng mô hình Decision Tree

decision\_tree\_model = DecisionTreeClassifier()

* Huấn luyện mô hình

decision\_tree\_model.fit(X\_train, y\_train)

* Lưu mô hình vào file

joblib.dump(decision\_tree\_model, "decision\_tree\_model.pkl")

## Random forest

### Cơ sở lý thuyết

* Random Forest là một phương pháp học máy dựa trên cả cây quyết định và sự kết hợp của chúng. Phương pháp này hoạt động bằng cách xây dựng một loạt các cây quyết định tại các điểm dữ liệu ngẫu nhiên, sau đó kết hợp kết quả từ các cây này để tạo ra dự đoán cuối cùng. Quan trọng nhất, Random Forest có khả năng giảm overfitting, một vấn đề thường gặp trong việc xây dựng các mô hình phức tạp.
* Thuật toán:
* Input: Bộ dữ liệu gồm n dữ liệu (sample) và mỗi dữ liệu có k thuộc tính (feature).
* Output: Dự đoán cho dữ liệu mới dựa trên tổng hợp kết quả từ các cây quyết định.
  + Khởi tạo: chọn số lượng cây quyết định và chọn ngẫu nhiên một tập con của dữ liệu huấn luyện cho mỗi cây.
* Xây dựng cây quyết định:
  + Đối với mỗi cây:
    - Chọn ngẫu nhiên một tập con của các thuộc tính.
    - Xây dựng cây quyết định bằng cách sử dụng tập con dữ liệu và tập con thuộc tính đã chọn.
* Tại mỗi nút trong cây:
  + Chọn thuộc tính tốt nhất để chia dữ liệu.
  + Chia dữ liệu thành hai phần dựa trên giá trị của thuộc tính được chọn.
  + Tiếp tục xây dựng cây con cho mỗi phần dữ liệu.
* Dự đoán:
  + Đối với mỗi cây: Dự đoán kết quả bằng cách đi qua cây từ gốc đến lá.
  + Đối với bài toán phân loại: Đưa ra dự đoán cuối cùng bằng cách bỏ phiếu.
* Kết quả cuối cùng: Kết quả đa số các cây.

### Vì sao chọn Random forest cho bài toán

Random Forest được ưa chuộng trong machine learning với khả năng giảm overfitting và tự động đánh giá độ quan trọng của các đặc trưng, đồng thời cung cấp khả năng xử lý dữ liệu lớn (cụ thể trong bài toán này là 69047 bản ghi) và linh hoạt cho bài toán.

### Áp dụng cơ sở lý thuyết vào xây dựng mô hình cho bài toán

* Đọc dữ liệu từ file CSV

data = pd.read\_csv('D:\kpdl\merged\_file1.csv')

* Phân chia dữ liệu thành 2 phần độc lập và phụ thuộc

X = data.loc[:, ['veil-color','habitat', 'ring-type', 'cap-shape', 'spore-print-color', 'cap-color', 'gill-attachment', 'veil-type','gill-color', "gill-spacing", "cap-surface"]]

y = data.loc[:, 'class']

* Chuyển đổi biến đầu vào thành giá trị nguyên

encoder = LabelEncoder()

for i in X.columns:

X[i] = encoder.fit\_transform(X[i])

y = encoder.fit\_transform(y)

* Chia dữ liệu thành tập training và tập test (theo tỷ lệ 7:3)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3)

* Tạo mô hình

logistic\_classifier\_model = LogisticRegression()

ridge\_classifier\_model = RidgeClassifier()

decision\_tree\_model = DecisionTreeClassifier()

naive\_bayes\_model = GaussianNB()

neural\_network\_model = MLPClassifier()

* Huấn luyện mô hình

logistic\_classifier\_model.fit(X\_train, y\_train)

ridge\_classifier\_model.fit(X\_train, y\_train)

decision\_tree\_model.fit(X\_train, y\_train)

naive\_bayes\_model.fit(X\_train, y\_train)

neural\_network\_model.fit(X\_train, y\_train)

import joblib

joblib.dump(decision\_tree\_model, "decision\_tree\_model.pkl")

* Test mô hình

tree\_pred = decision\_tree\_model.predict(X\_test)

# ĐÁNH GIÁ MÔ HÌNH HỌC MÁY

## Kiểm tra chéo đánh giá hiệu suất 2 mô hình (thực hiện kiểm tra chéo 10 lần)

1.1. Mô hình Decision tree

* Thuật toán:

from sklearn.model\_selection import cross\_val\_score

import pandas as pd

# Cross-validation for Decision Tree

decision\_tree\_scores = cross\_val\_score(decision\_tree\_model, X, y, cv=10, scoring='accuracy')

# Tạo DataFrame từ kết quả kiểm tra chéo của Decision Tree

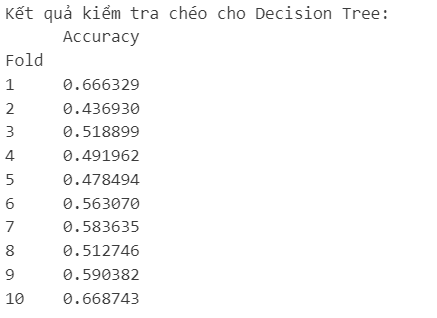
decision\_tree\_df = pd.DataFrame({'Fold': range(1, 11), 'Accuracy': decision\_tree\_scores})

decision\_tree\_df.set\_index('Fold', inplace=True)

print("\nKết quả kiểm tra chéo cho Decision Tree:")

print(decision\_tree\_df)

* Kết quả:



1.2. Mô hình Random Forest

* Thuật toán:

from sklearn.model\_selection import cross\_val\_score

import pandas as pd

# Cross-validation for Random Forest

random\_forest\_scores = cross\_val\_score(random\_forest\_model, X, y, cv=10, scoring='accuracy')

# Tạo DataFrame từ kết quả kiểm tra chéo của Random Forest

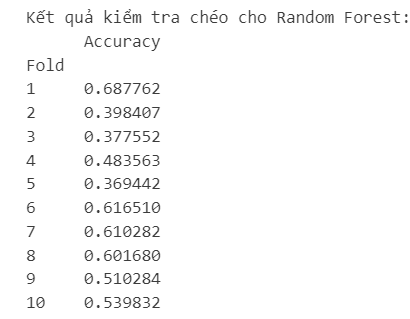
random\_forest\_df = pd.DataFrame({'Fold': range(1, 11), 'Accuracy': random\_forest\_scores})

random\_forest\_df.set\_index('Fold', inplace=True)

print("\nKết quả kiểm tra chéo cho Random Forest:")

print(random\_forest\_df)

* Kết quả:



## Đánh giá khi chạy mô hình test và mô hình train

## 2.1 Decision tree

# Dự đoán và đánh giá trên tập kiểm tra

tree\_pred = decision\_tree\_model.predict(X\_test)

tree\_report\_test = classification\_report(y\_test, tree\_pred)

print('Decision Tree - Test')

print(tree\_report\_test)

# Đánh giá trên tập huấn luyện

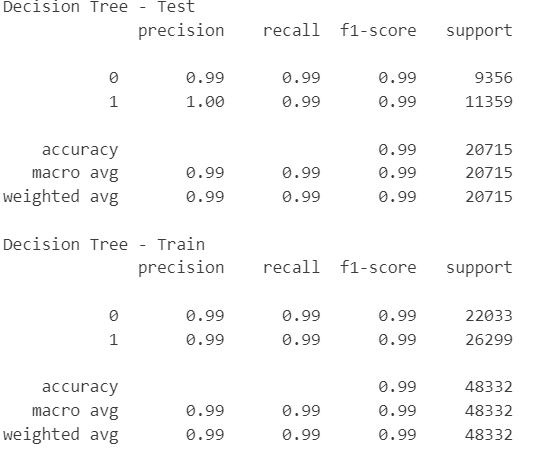
tree\_pred\_train = decision\_tree\_model.predict(X\_train)

tree\_report\_train = classification\_report(y\_train, tree\_pred\_train)

print('Decision Tree - Train')

print(tree\_report\_train)

Kết quả:



## 2.2 Random Forest

# Dự đoán và đánh giá trên tập kiểm tra

random\_forest\_pred = random\_forest\_model.predict(X\_test)

random\_forest\_report\_test = classification\_report(y\_test, random\_forest\_pred)

print('Random Forest - Test')

print(random\_forest\_report\_test)

# Đánh giá trên tập huấn luyện

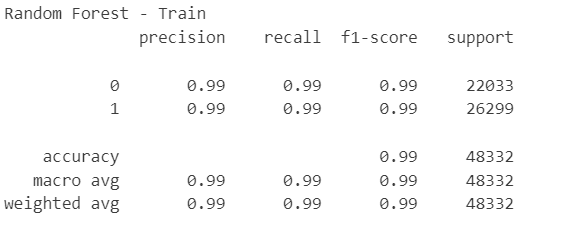
random\_forest\_pred\_train = random\_forest\_model.predict(X\_train)

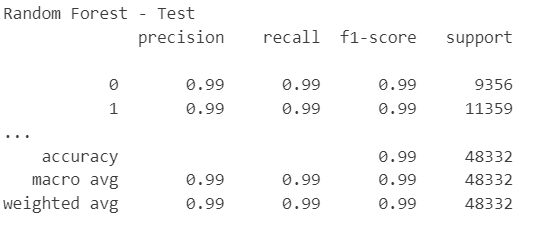
random\_forest\_report\_train = classification\_report(y\_train, random\_forest\_pred\_train)

print('Random Forest - Train')

print(random\_forest\_report\_train)

Kết quả:





## Đánh giá mô hình thông qua ma trận nhầm lẫn (Confusing Matrix).

* Ma trận nhầm lẫn (confusion matrix) là công cụ trực quan và hiệu quả để đánh giá hiệu suất của mô hình phân loại. Nó thể hiện rõ ràng mức độ chính xác của mô hình trong việc dự đoán các lớp (classes) khác nhau.
* Phương pháp đánh giá:
  + - Tính toán các thước đo
      * Accuracy: tỉ lệ giữa số điểm dữ liệu được dự đoán đúng và tổng số điểm dữ liệu.
      * Precision: tỉ lệ số điểm true positive trong số những điểm được phân loại là positive
      * Recall: tỉ lệ số điểm true
      * positive trong số những điểm thực sự là positive
      * F1 Score: Điểm cân bằng giữa precision và recall.
      * TPR
      * FPR
    - Phân tích các thước đo trong ma trận: Một mô hình tốt khi cả Precision và Recall đều cao có nghĩa là ít phân loại nhầm và tỷ lệ bỏ sót thấp
* Công thức tính:
  + - Accuracy= (TP+TN)/(TP+TN+FP+FN)
    - Precision= TP/(TP+ FP)
    - Recall= TP/(TP+ FN)
    - Accuracy=2\* (Precision\*Recall )/(Precision+Recall)
    - TPR = TP / (TP + FN)
    - FPR = FP / (FP + TN)

Trong đó:

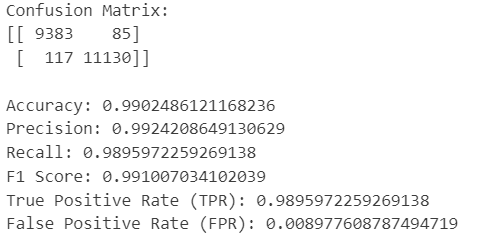
TP( True Positive): Số lượng các trường hợp mà mô hình dự đoán đúng là positive (dương tính) và thực tế cũng là positive.

TN( True Negative): Số lượng các trường hợp mà mô hình dự đoán đúng là negative và thực tế cũng là negative.

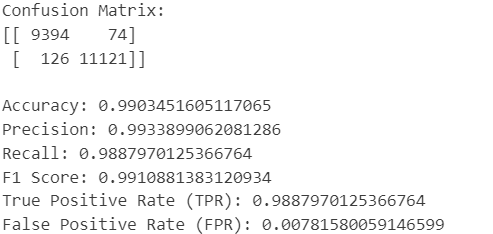
FP( False Positive): Số lượng các trường hợp mà mô hình dự đoán là positive nhưng thực tế lại là negative (âm tính).

FN( False Negative): Số lượng các trường hợp mà mô hình dự đoán là negative nhưng thực tế lại là positive.

* Đánh giá mô hình Random forest



* + Nhận xét:
    - Mô hình có Accuracy cao (99,02%)
    - Mô hình có Precision cao (99,24%)
    - Mô hình có Recall rất cao (98,95%)
    - Mô hình có: F1 Score cao (99,10%)
    - Mô hình có TPR cao (98,95%)
    - Mô hình có FPR thấp (0,89%)
      * Mô hình có hiệu suất tốt
* Đánh giá mô hình Decision Trees:



* + Nhận xét:
    - Mô hình có Accuracy cao (99,03%)
    - Mô hình có Precision cao (99,33%)
    - Mô hình có Recall rất cao (98,87%)
    - Mô hình có: F1 Score cao (99,10%)
    - Mô hình có TPR cao (98,87%)
    - Mô hình có FPR thấp (0.78%)
      * Mô hình có hiệu suất tốt
* Nhận xét:

Cả hai mô hình Random Forest và Decision Tree đều có hiệu suất rất tốt với các độ đo như Accuracy, Precision, Recall, F1 Score, True Positive Rate (TPR), và False Positive Rate (FPR) gần như tương đương. Một số điểm khác biệt nhỏ giữa hai mô hình:

* Random Forest có FPR cao hơn so với Decision Tree (0,89% so với 0,78%). Điều này có thể chỉ ra rằng Random Forest có một chút cần cải thiện hơn trong việc tránh dự đoán sai (false positive) so với Decision Tree.
* Decision Tree có Precision cao hơn so với Random Forest (99,33% so với 99,24%). Điều này có nghĩa là Decision Tree ít dự đoán nhãn positive sai hơn so với Random Forest

Tuy nhiên, thời gian chạy thuật toán của random forest là 1,9s còn Decisiontree là 0,0s Tree

* + - * Mô hình Decision Tree có hiệu suất tốt hơn cho bài toán đặt ra

# TÍCH HỢP MÔ HÌNH HỌC MÁY VÀO ỨNG DỤNG

* Thuật toán

import tkinter as tk

from PIL import Image, ImageTk

import joblib

*# Load mô hình đã được huấn luyện*

def load\_model():

*# Thực hiện việc load mô hình đã được huấn luyện từ file hoặc từ bộ nhớ*

*# Trong ví dụ này, ta giả định mô hình đã được huấn luyện và lưu trong biến clf*

clf = joblib.load("decision\_tree\_model.pkl")

*# Đoạn này có thể thay đổi tùy theo cách bạn lưu và load mô hình*

return clf

*# Dự đoán kết quả khi nhấn nút*

def on\_select(event):

selected\_option = []

for i in range(11):

selected\_option.append(variables[i].get())

print("Selected options:", selected\_option)

*# Hàm được gọi khi nút "Dự đoán" được nhấn*

def predict():

selected\_options = []

for var in variables:

selected\_option\_name = var.get()

selected\_option\_value = options\_lists[variables.index(var)][selected\_option\_name]

selected\_options.append(selected\_option\_value)

prediction = clf.predict([selected\_options])

result\_str = ""

icon\_path = ""

if prediction[0] == 1:

result\_str = "Nấm độc"

icon\_path = "poison.png" *# Đường dẫn đến biểu tượng cho nấm độc*

else:

result\_str = "Nấm ăn được"

icon\_path = "shield.png" *# Đường dẫn đến biểu tượng cho nấm ăn được*

result\_label.config(text=f"Loại nấm này là: {result\_str}")

*# Hiển thị biểu tượng*

load\_icon(icon\_path)

*# Tạo giao diện*

root = tk.Tk()

root.title("Dự đoán Loại Nấm")

*# Tạo Frame chứa các widget để căn giữa*

frame = tk.Frame(root)

frame.pack(expand=True, fill='both') *# Frame mở rộng và điền đầy toàn bộ root*

*# Tạo các ô nhập liệu cho 11 thuộc tính*

title = ['veil-color', 'habitat', 'ring-type', 'cap-shape', 'spore-print-color', 'cap-color', 'gill-attachment',

'veil-type', 'gill-color', "gill-spacing", "cap-surface"]

options\_lists = [

{"orange(0)": 0, "brown(1)": 1, "purple(2)": 2, "unknown(3)": 3, "white(4)": 4, "yellow(5)": 5, "black(6)": 6,

"red(7)": 7}, *# veil color*

{"grasses(0)": 0, "heaths(1)": 1, "urban(2)": 2, "woods(3)": 3, "leaves(4)": 4, "paths(5)": 5, "waste(6)": 6,

"meadows(7)": 7}, *# habitat*

{"flaring(0)": 0, "grooved(1)": 1, "none(2)": 2, "none'(3)": 3, "unknown(4)": 4, "zone(5)": 5, "large(6)": 6,

"pendant(7)": 7, "movable(8)": 8, "evanescent(9)": 9}, *# ring-type*

{"sunken(0)": 0, "others(1)": 1, "flat(2)": 2, "convex(3)": 3, "knobbed(4)": 4, "bell(5)": 5, "spherical(6)": 6,

"conical(7)": 7}, *# cap-shape*

{"green(0)": 0, "gray(1)": 1, "yellow(2)": 2, "orange(3)": 3, "chocolate(4)": 4, "brown(5)": 5, "purple(6)": 6,

"unknown(7)": 7, "white(8)": 8, "black(9)": 9, "buff(10)": 10, "pink(11)": 11},

{"green(0)": 0, "gray(1)": 1, "orange(2)": 2, "brown(3)": 3, "purple(4)": 4, "blue(5)": 5, "white(6)": 6,

"yellow(7)": 7, "pink(8)": 8, "buff(9)": 9, "black(10)": 10, "red(11)": 11, "cinnamon(12)": 12}, *# cap-color*

{"adnexed(0)": 0, "sinuate(1)": 1, "decurrent(2)": 2, "unknown(3)": 3, "none(4)": 4, "adnexed(5)": 5, "pores(6)": 6,

"free(7)": 7}, *# gill-attachment*

{"unknown(0)": 0, "universial(1)": 1, "partial(2)": 2}, *# veil-type*

{"green(0)": 0, "gray(1)": 1, "yellow(2)": 2, "orange(3)": 3, "brown(4)": 4, "chocolate(5)": 5, "purple(6)": 6,

"none(7)": 7, "pink(8)": 8, "white(9)": 9, "black(10)": 10, "buff(11)": 11, "red(12)": 12}, *# gill-color*

{"distant(0)": 0, "unknown(1)": 1, "none(2)": 2, "crowded(3)": 3, "close(4)": 4},

{"smooth(0)": 0, "grooves(1)": 1, "shiny(2)": 2, "fibrous(3)": 3, "d(4)": 4, "fibrous'(5)": 5, "unknown(6)": 6,

"leathery(7)": 7, "wrinkled(8)": 8, "scaly(9)": 9, "silky(10)": 10, "sticky(11)": 11, "fleshy(12)": 12}

*# cap-surface*

]

option\_values = [{name: value for value, name in options.items()} for options in options\_lists]

*# Biến lưu trữ giá trị được chọn cho mỗi dropdown*

variables = []

for i in range(11):

options = list(options\_lists[i].keys())

variable = tk.StringVar(root)

variable.set(options[0]) *# Giá trị mặc định cho dropdown thứ i*

variables.append(variable)

*# Tạo nhãn*

label = tk.Label(frame, text=f"{title[i]}:")

label.grid(row=i, column=0, padx=5, pady=5, sticky='w')

*# Tạo dropdown list*

dropdown = tk.OptionMenu(frame, variable, \*options, command=on\_select)

dropdown.grid(row=i, column=1, padx=5, pady=5, sticky='w')

*# Tải mô hình đã được huấn luyện*

clf = load\_model()

*# Biến lưu trữ biểu tượng*

icon\_label = None

*# Hàm để tải biểu tượng và hiển thị*

def load\_icon(icon\_path):

global icon\_label

*# Xóa biểu tượng cũ trước khi hiển thị biểu tượng mới*

if icon\_label:

icon\_label.grid\_forget()

*# Load biểu tượng từ file hình ảnh*

icon = Image.open(icon\_path)

icon = icon.resize((30, 30))

icon = ImageTk.PhotoImage(icon)

*# Hiển thị biểu tượng*

icon\_label = tk.Label(frame, image=icon)

icon\_label.image = icon *# Giữ tham chiếu tới hình ảnh để không bị thu hồi bộ nhớ*

icon\_label.grid(row=13, columnspan=2, padx=5, pady=5, sticky='e')

*# Nút "Dự đoán"*

predict\_button = tk.Button(frame, text="Dự đoán", command=predict)

predict\_button.grid(row=12, columnspan=1, padx=5, pady=5)

*# Hàm để reset kết quả dự đoán và thiết lập lại các dropdown về trạng thái mặc định*

def reset():

result\_label.config(text="")

if icon\_label:

icon\_label.grid\_forget()

for var in variables:

var.set(list(options\_lists[variables.index(var)].keys())[0]) *# Thiết lập dropdown về giá trị mặc định*

*# Nút "Reset"*

reset\_button = tk.Button(frame, text="Làm mới", command=reset)

reset\_button.grid(row=12, column=1, padx=5, pady=5)

*# Kết quả dự đoán*

result\_label = tk.Label(frame, text="", justify='left')

result\_label.grid(row=13, columnspan=1, padx=5, pady=5, sticky='w')

*# Căn giữa frame*

root.update\_idletasks()

frame\_width = frame.winfo\_reqwidth()

frame\_height = frame.winfo\_reqheight()

screen\_width = root.winfo\_screenwidth()

screen\_height = root.winfo\_screenheight()

x = (screen\_width - frame\_width) // 2

y = (screen\_height - frame\_height) // 2

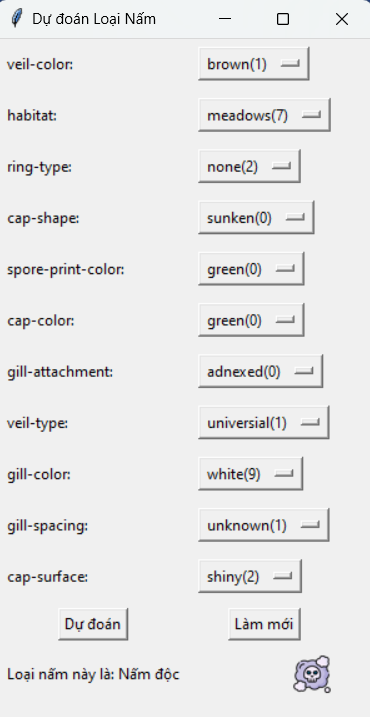
root.geometry("+{}+{}".format(x, y))

*# Khởi chạy giao diện*

root.geometry("300x600+{}+{}".format(x, y))

root.mainloop()

* Kết quả



# KẾT LUẬN

Trong quá trình làm bài tập lớn, nhóm đã thực hiện các công việc sau:

* Nghiên cứu về các phương pháp học máy có giám sát.
* Áp dụng các giai đoạn của chu trình học máy vào bài toán thực tế
* Thực hiện đánh giá, so sánh 2 mô hình học để lựa chọn mô hình học máy tốt nhất
* Tích hợp được mô hình học máy vào ứng dụng để giải quyết các vấn đề trong lĩnh vực nghiên cứu y tế.

**TÀI LIỆU THAM KHẢO**

[1] Bài giảng môn Nhập môn Khai phá dữ liệu và Máy học, TS Vũ Xuân Hạnh, Khoa Công nghệ thông tin, Trường Đại Học Mở Hà Nội

[2] Machine Learning cơ bản, Vũ Hữu Tiệp

[3]<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.drop_duplicates.html>

[4] <https://github.com/moluchase/deeplearning_demo>

[5] <https://www.geeksforgeeks.org/python-pandas-dataframe-drop_duplicates/>

[6] <https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html>

[7] Bài giảng môn Nhập môn Khai phá dữ liệu và Máy học, Nguyễn Nhật Quang, Đại Học Bách Khoa Hà Nội <https://drive.google.com/file/d/1_Gsz68GAulRJUzaGIgq93B70PVHYSuPC/view>

[8] [Thuật toán rừng ngẫu nhiên trong Machine Learning - GeeksforGeeks](https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/)