# **LỜI MỞ ĐẦU**

Ngày nay, với cuộc cách mạng 4.0 cùng sự bùng nổ và phát triển của mạng 5G, AI (Artificial Intelligence)hay còn gọi là Trí tuệ nhân tạo đã trở thành một ngành khoa học máy tính được nhiều công ty công nghệ lớn trên thế giới rất quan tâm phát triển. Trí tuệ nhân tạo là một bộ phận của khoa học máy tính liên quan đến việc tự động hóa và thông minh, do đó nó được xây dựng dựa trên những nguyên lý lý thuyết vững chắc, có khả năng ứng dụng cao len lỏi vào mọi lĩnh vực trong đời sống chúng ta.

Hầu hết mọi ngành công nghệ đang làm việc với lượng dữ liệu lớn ngày nay đều nhận ra tầm quan trọng của học máy, bởi học máy là một phương pháp để huấn luyện dữ liệu, giúp máy tính trở nên thông minh hơn thông qua các phương pháp học và cụ thể hơn là các thuật toán. Công nghệ máy học hỗ trợ nhiều khía cạnh của xã hội hiện đại: từ tìm kiếm trên web đến lọc nội dung trên mạng xã hội đến các đề xuất trên các trang web thương mại điện tử và nó ngày càng hiện diện trong các sản phẩm tiêu dùng như máy ảnh và điện thoại thông minh. Hệ thống máy học được sử dụng để xác định các đối tượng trong hình ảnh, chuyển lời nói thành văn bản, khớp các mục tin tức, bài đăng hoặc sản phẩm với sở thích của người dùng và chọn các kết quả tìm kiếm có liên quan.

Càng ngày, các ứng dụng này càng sử dụng một lớp kỹ thuật được gọi là học sâu. Vậy nên, các ứng dụng của học máy đã dần trở nên quen thuộc với mọi người: xe tự hành của Google và Tesla, hệ thống tự tag khuôn mặt trên Facebook, hệ thống gợi ý sản phẩm của Amazon, …. Sau một thời gian tìm hiểu về học máy với mong muốn áp dụng được những kiến thức đã học vào thực tiễn, chúng em đã quyết định chọn đề tài: “Kiến trúc mạng Darknet và ứng dụng YOLO.”

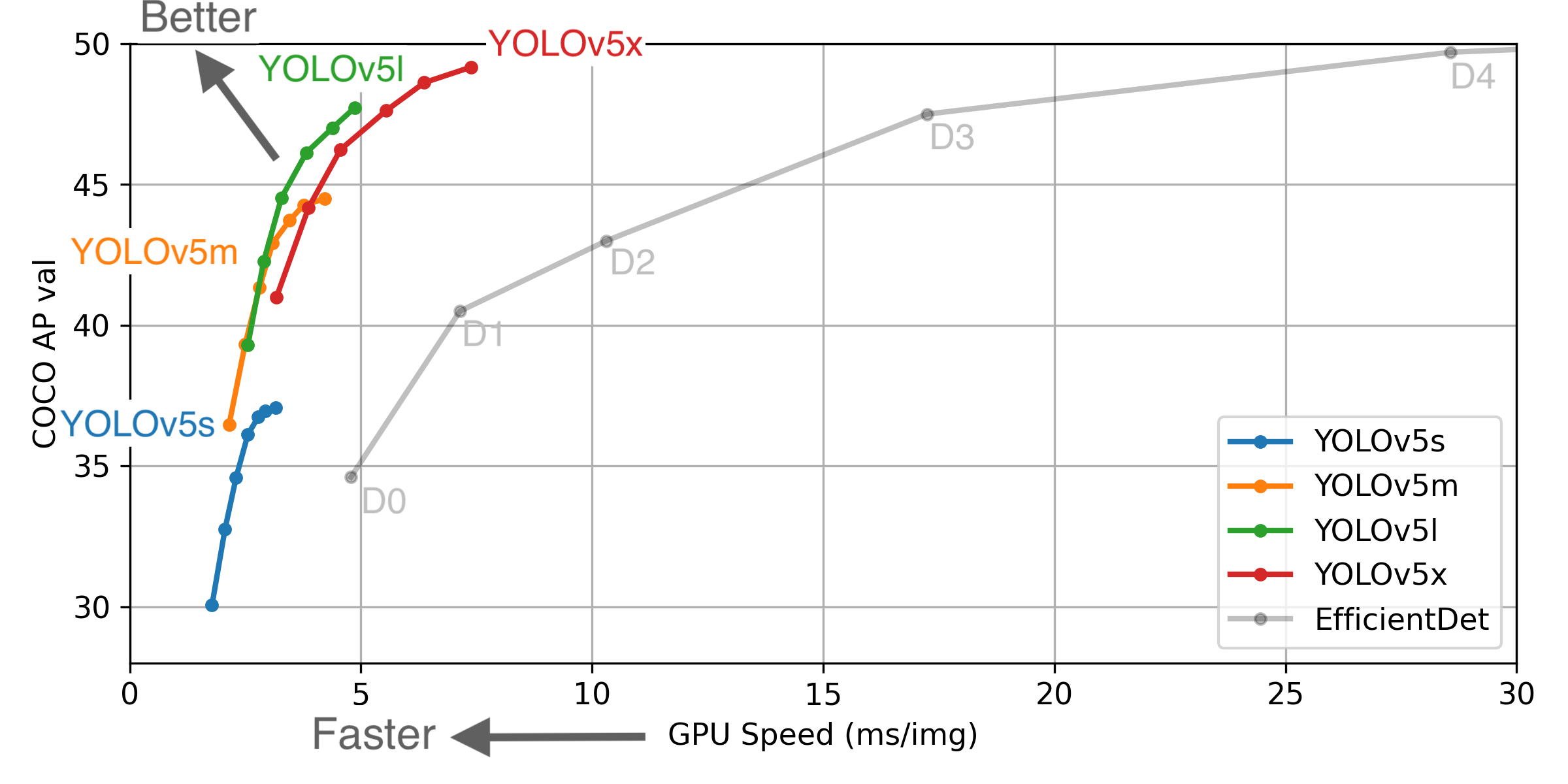
# **CHƯƠNG I: KIẾN TRÚC MẠNG DARKNET VÀ ỨNG DỤNG YOLO TRONG OBJECT DETECTION**

## **1.1. Giới thiệu về kiến trúc mạng Darknet**

* Darknet là một khung mạng thần kinh mã nguồn mở được viết bằng C và CUDA. Nó nhanh chóng, dễ cài đặt và hỗ trợ tính toán CPU và GPU.

## **1.2. Giới thiệu về Yolo và cách cài đặt**

Object Detection là một bài toán quan trọng trong lĩnh vực Computer Vision, thuật toán Object Detection được chia thành 2 nhóm chính:

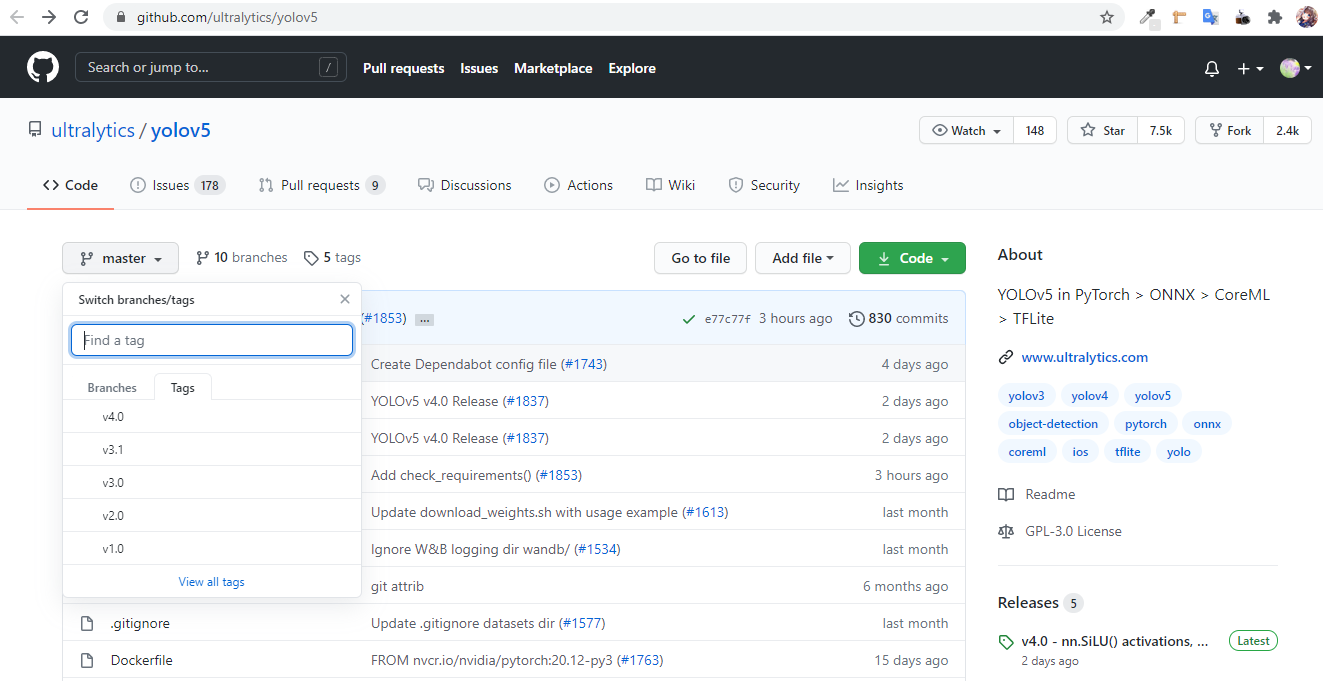


* Họ các mô hình RCNN (Region-Based Convolutional Neural Networks) để giải quyết các bài toán về định vị và nhận diện vật thể.
* Họ các mô hình về YOLO (You Only Look Once) dùng để nhận dạng đối tượng được thiết kế để nhận diện các vật thể real-time.

Yolo là một mô hình mạng CNN cho việc phát hiện, nhận dạng, phân loại đối tượng. Yolo được tạo ra từ việc kết hợp giữa các convolutional layers và connected layers.Trong đó các convolutional layers sẽ trích xuất ra các feature của ảnh, còn full-connected layers sẽ dự đoán ra xác suất đó và tọa độ của đối tượng.

**1.3. Cài đặt Yolo**

- Bước 1: Truy cập link <https://github.com/ultralytics/yolov5> , sau đó chọn master → tags. Giờ bạn có thể chọn phiên bản ổn định nhất để tải về, ở đây chúng tôi chọn tải Yolov5- v3.1

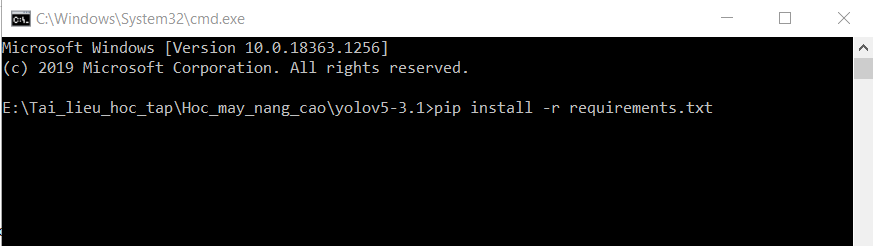


- Bước 2: Cài đặt môi trường

Cài đặt môi trường Python >=3.5, ta truy cập link https://www.python.org/downloads/

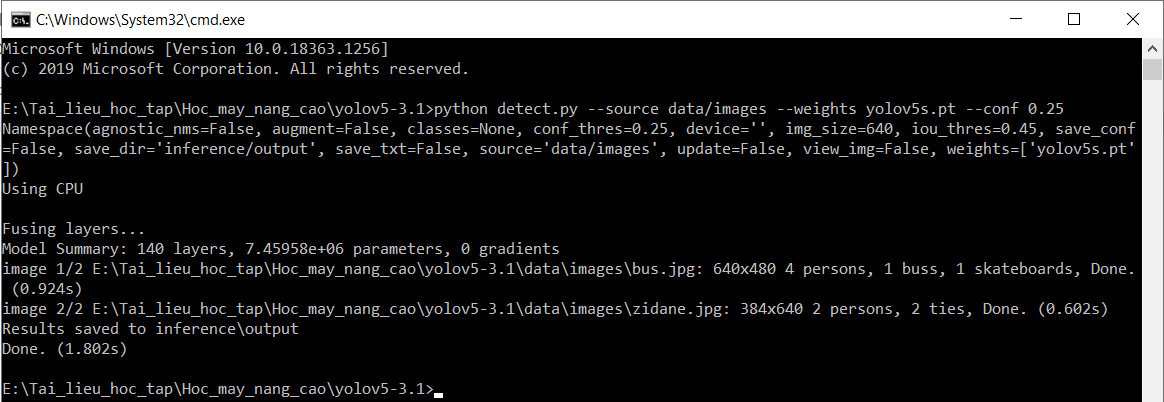
cmd đến thu mục Yolov5-3.1 bạn vừa tải và gõ:

pip install -r requirements.txt



- Bước 3: Chạy phát hiện đối tượng với mô hình có sẵn ta chạy dòng lệnh:

python detect.py --source data/images --weights yolov5s.pt --conf 0.25

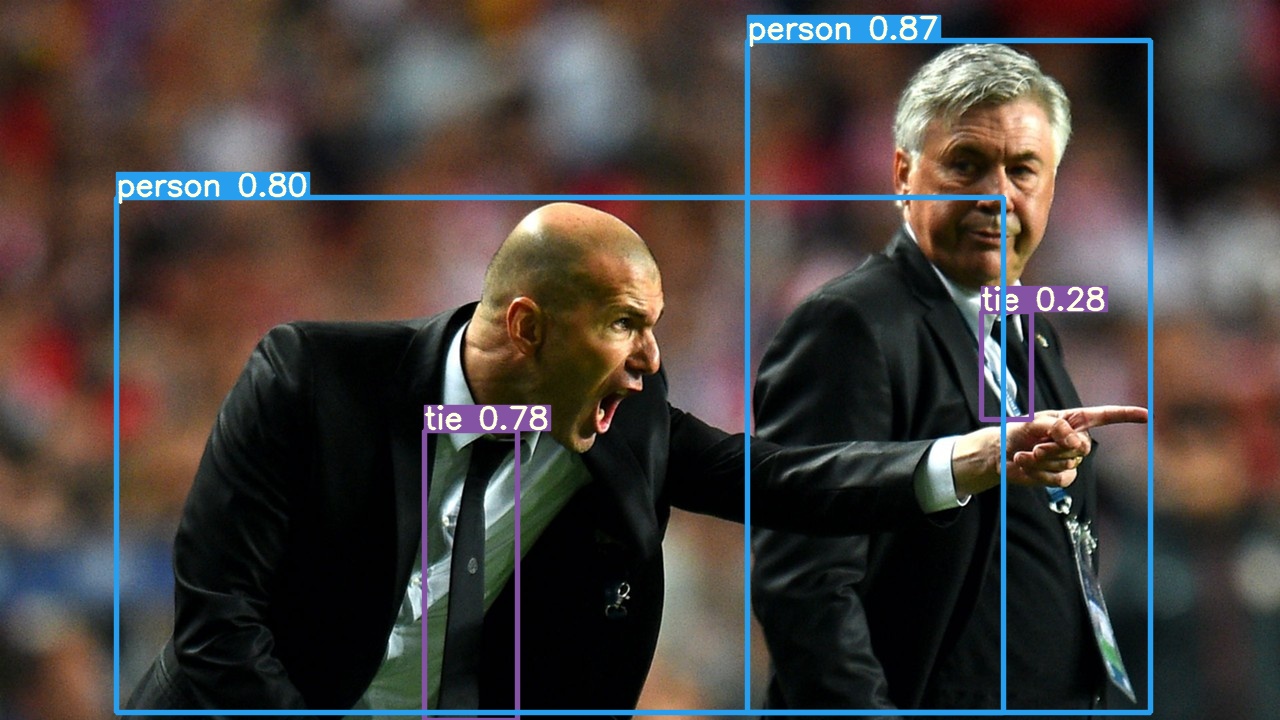


Kết quả:

Ảnh trước khi chạy:



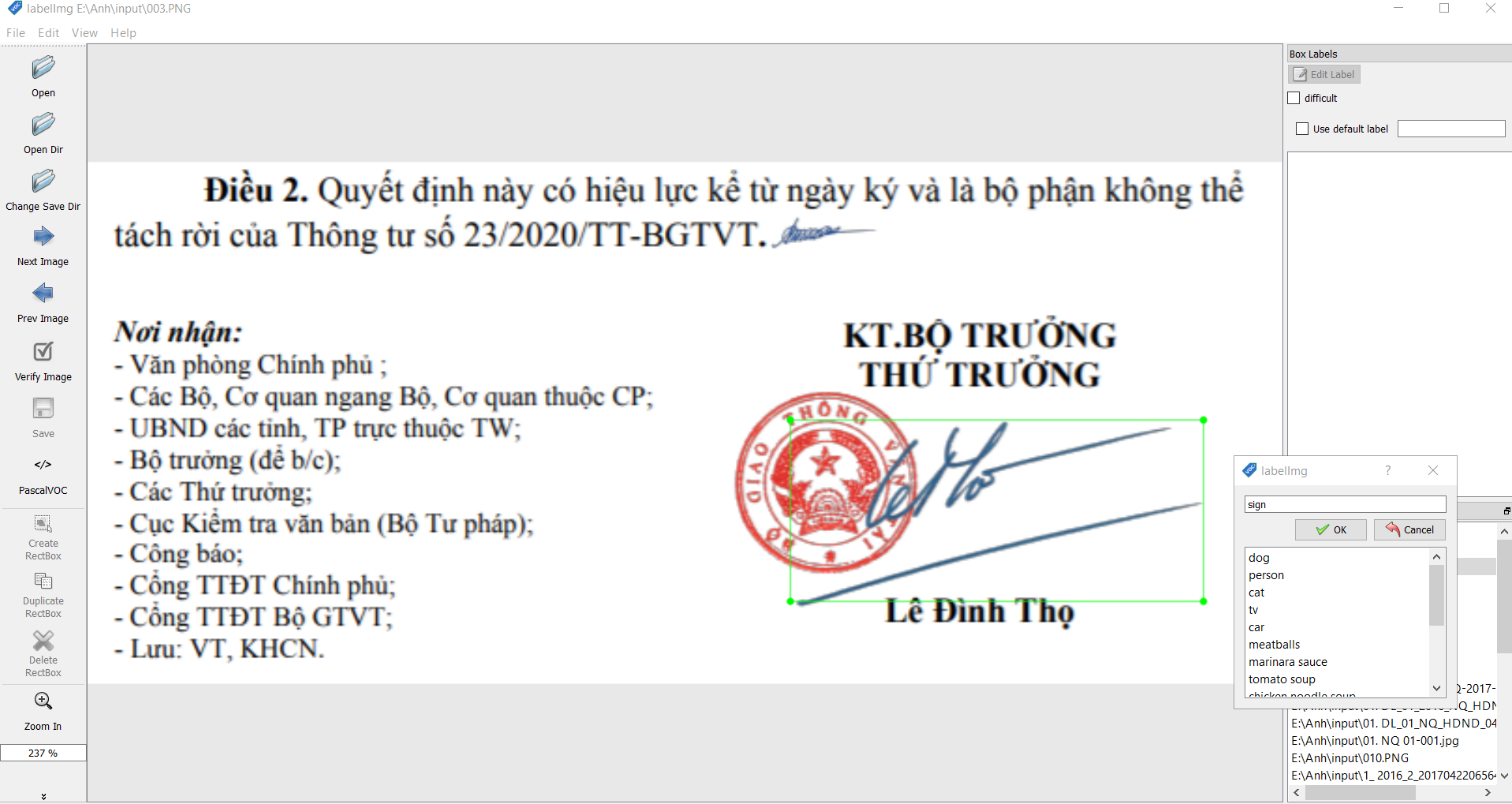
Ảnh sau khi chạy:



**1.4. Huấn luyện phát hiện chữ ký với Yolov5**

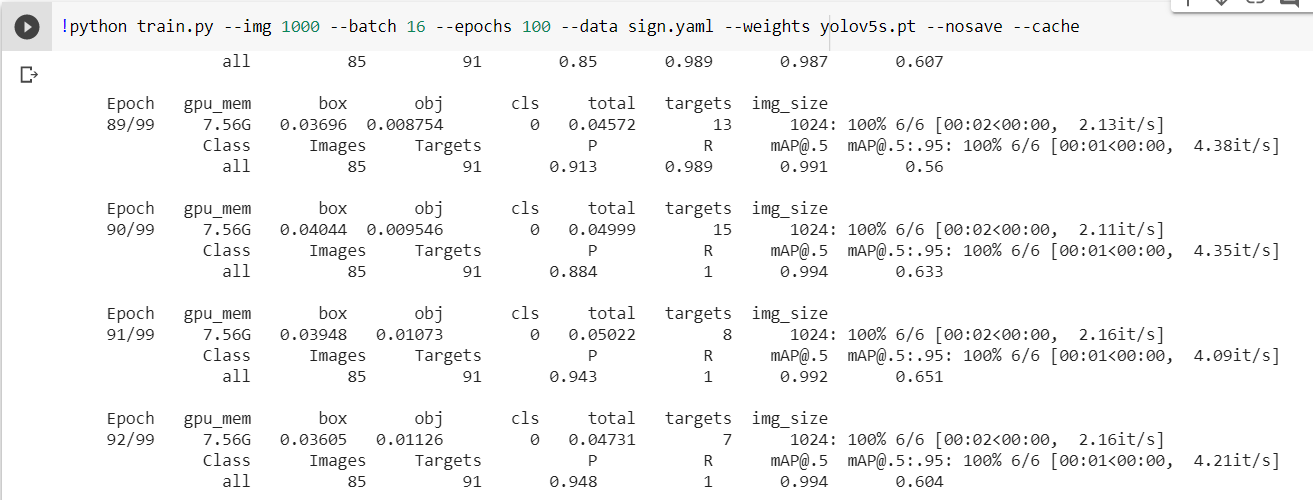
* Chuẩn bị dữ liệu và làm nhãn:

Link tải: https://tzutalin.github.io/labelImg/



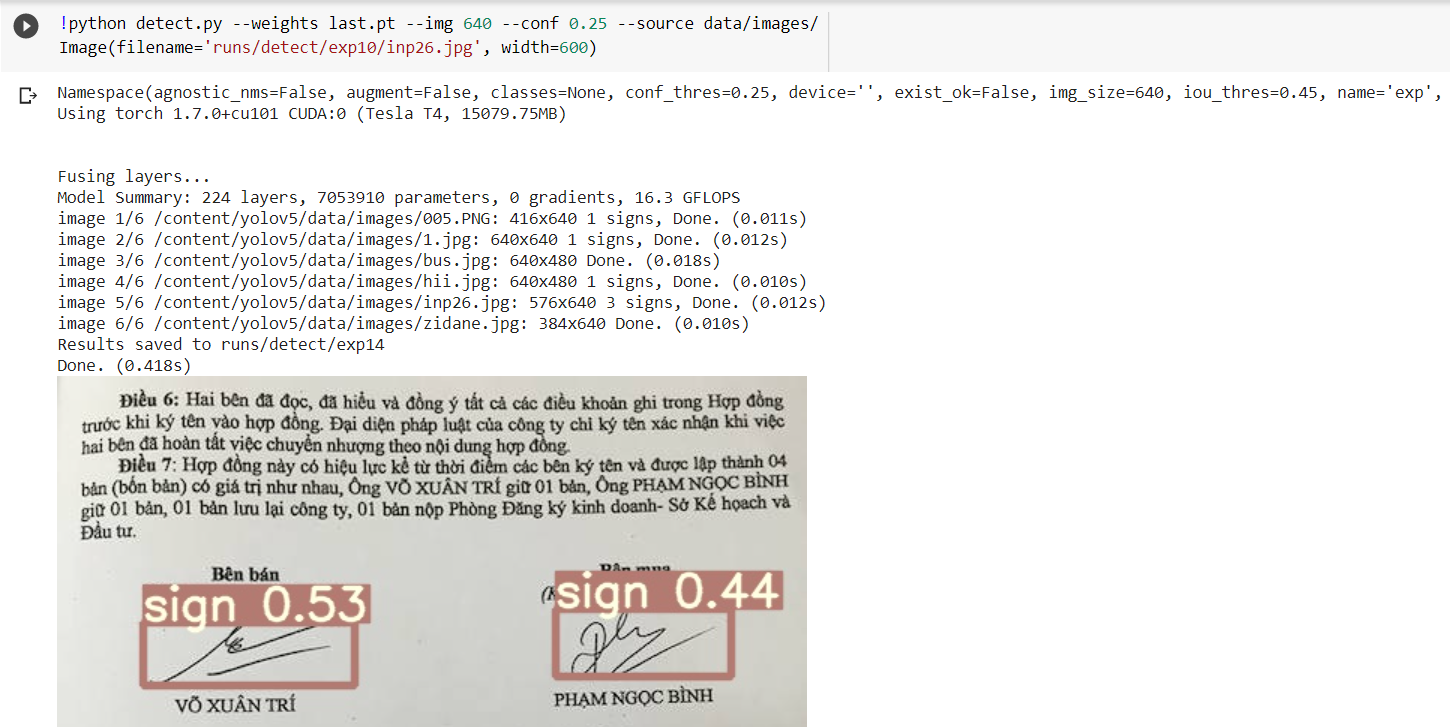
* Train trên Colab ta chạy dòng lệnh:

!python train.py --img 1000 --batch 16 --epochs 100 --data sign.yaml --weights yolov5s.pt --nosave --cache

****

Sau khi train ta được một file last.pt

* Kết quả:

****

# **CHƯƠNG II. GIẢI CÁC BÀI TOÁN HỌC MÁY**

## **2.1 Bài toán phân cụm nội dung tin nhắn giống nhau với bộ dữ liệu tweets**

### **2.1.1 Mô tả bài toán**

Twitter cung cấp dịch vụ đăng tin nhắn ngắn. Trong thực tế, nhiều tweet rất giống nhau và có thể gom lại được với nhau. Bằng cách nhóm các tweet tương tự lại với nhau, chúng ta có thể tạo ra một bảng trình bày ngắn gọn và có tổ chức hơn cho các tweet thô, điều này sẽ rất hữu ích cho nhiều ứng dụng dựa trên twitter (ví dụ: khám phá sự thật, phân tích xu hướng, xếp hạng tìm kiếm.

Ở đây, các tweet được phân nhóm bằng cách sử dụng thước đo khoảng cách Jaccard và thuật toán K – Mean

Khoảng cách Jaccard đo lường sự không giống nhau giữa hai bộ mẫu (A và B). Nó được định nghĩa là hiệu số của kích thước của hợp và giao của hai tập hợp chia cho kích thước của hợp của các tập hợp.

Lưu ý bạn có thể sử dụng các giải thuật khác để tìm điểm tương đồng giữa nội dung văn bản, Jaccard chỉ có thể phân biệt được 70% nội dung đưa vào.

Dist(A,B) = 1 - |A giao B| / |A union B|

Ví dụ: Hãy xem xét các tweet sau:

Tweet A: Cuộc hành trình dài.

Tweet B: Cuộc sống dài.

|A giao B| = 1 và |A union B| = 5, do đó khoảng cách là 1 – (1/5)

Khoảng cách Jaccard Dist (A,B) giữa tweet A và B có các thuộc tính sau:

1. Thật nhỏ nếu tweet A và B tương tự nhau.

2. Nó là lớn nếu chúng không giống nhau.

3. Nó là 0 nếu chúng giống nhau.

4. Nó là 1 nếu chúng hoàn toàn khác nhau (tức là không có từ trùng lặp).

Thực hiện bài toán:

- Bài toán phân cụm

- Sử dụng kỹ thuật K – Mean, Jaccard

- Tập dữ liệu sử dụng trong thuật toán: Health News in Twitter. Thông tin đầy đủ của tập dữ liệu gồm:

o 3929 dòng dữ liệu

o Id tweet

o Ngày và giờ đăng tweet

o Nội dung tweet

Link tải dataset: <https://archive.ics.uci.edu/ml/datasets/Health+News+in+Twitter>

### **2.1.2 Tiền xử lý dữ liệu**

Các tweet trước khi đưa vào thuật toán sẽ phải tiền xử lý dữ liệu gồm các bước sau:

- Xóa dấu Id và thời gian tweet.

- Các từ bắt đầu bằng @, ví dụ @XuanNam sẽ bị bỏ đi.

- Các biểu tượng thẻ bắt đầu bằng # bị xóa, ví dụ #XuanNam sẽ bị bỏ đi.

- Tất cả Url sẽ bị xóa.

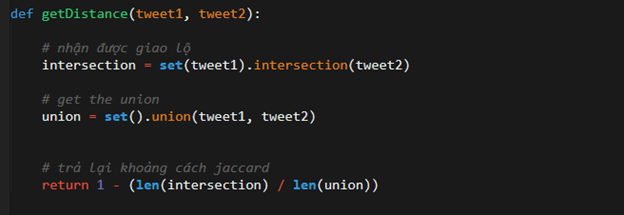
- Chuyển toàn bộ chữ thành chữ thường.

### **2.1.3 Các bước thực hiện**

Bước 1: Đầu tiên xây dựng giải thuật Jaccard.

- Đầu vào sẽ là 2 mảng tweet.

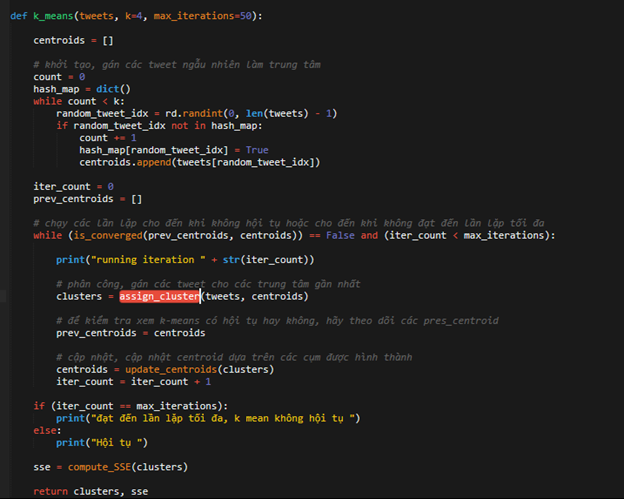
- Đầu ra là độ trùng hợp của 2 mảng tweet (0 -> 1)



Bước 2: Xây dựng thuật toán K – Mean.

- Đầu vào: Số tâm cụm, mảng tweet cần phân cụm, số lần lặp để tìm tâm cụm.

- Đầu ra: mảng tweet đã được đánh dấu tâm cụm tương ứng.



## **2.2. Bài toán phân lớp nhị phân dự đoán nấm ăn được hay có độc bằng kỹ thuật Logistic Regression**

### **2.2.1 Mô tả tập dữ liệu**

Tập dữ liệu gồm 8416 bản ghi với 23 cột dữ liệu

Bộ dữ liệu này bao gồm các mô tả về các mẫu giả định tương ứng với 23 loài nấm trong họ Agaricus và Lepiota. Mỗi loài được xác định là chắc chắn ăn được, chắc chắn có độc. Hướng dẫn nêu rõ rằng không có quy tắc đơn giản nào để xác định khả năng ăn được của nấm.

Các thuộc tính được mô tả như dưới đây

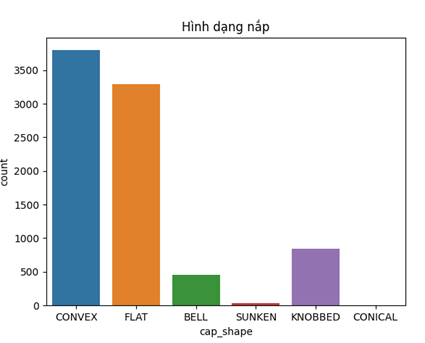
|  |  |  |
| --- | --- | --- |
| **STT** | **Tên** | **Mô tả** |
| 1 | cap\_shape | Hình dạng nắp: (convex:lồi lõm ,flat: phẳng ,knobbed: có núm ,bell: chuông , sunken: sâu thỏm , conical: hình nón ) |
| 2 | cap\_surface | Bề mặt nắp (scaly: có vảy , smooth: trơn tru , fibrous: dạng sợi , grooves: rãnh) |
| 3 | cap\_color | Màu nắp (brown: nâu, gray: xám, red:đỏ, yellow: vàng, white: trắng, buff: màu da, pink: hồng, cinnamon: quế, purple: tím, green:xanh lá) |
| 4 | bruises | Vết đốm (no: không có, bruises: có vết đốm) |
| 5 | odor | Mùi (none: không mùi, foul: hôi, spicy: cay, fishy: tanh, almond: mùi hạnh nhân, anise: mùi hồi, pungent: mùi hăng, creosote: mùi dầu creosote, musty: mùi mốc) |
| 6 | gill\_attachment | Chẻ nấm đính kèm (free: không, attached: có đính kèm) |
| 7 | gill\_spacing | Khoảng cách chẻ nấm (close: gần, crowded: chen chúc) |
| 8 | gill\_size | Kích thước chẻ nấm (broad: rộng lớn, narrow: hẹp) |
| 9 | gill\_color | Màu sắc chẻ nấm (buff: màu da, pink: hồng, white: trắng, brown: nâu, chocolate: màu socola, gray: xám tro, purple: tím, black: đen, red: đỏ, yellow: vàng, orange: cam, green: xanh lá) |
| 10 | stalk\_shape | Hình dạng cuống (tapering: hình nón, enlargin: mở rộng) |
| 11 | stalk\_root | Gốc thân cây (bulbous: có hình củ hành, club: chùm, rooted: vững chắc, equal: cân đều nhau) |
| 12 | stalk\_surface\_above\_ring | Bề mặt cuống trên vòng (smooth: trơn tru, silky: mướt, fibrous: có sợi, thớ; scaly: có vảy) |
| 13 | stalk\_surface\_below\_ring | Bề mặt cuống bên dưới vòng (smooth: trơn tru, silky: mướt, fibrous: có sợi, thớ; scaly: có vảy) |
| 14 | stalk\_color\_below\_ring | Màu cuống bên dưới vòng (buff: màu da, pink: hồng, white: trắng, brown: nâu, black: đen, red: đỏ, yellow: vàng, orange: cam, green: xanh lá) |
| 15 | veil\_type | Loại mạng che mặt: từng phần |
| 16 | veil\_color | Màu màn che (white: trắng, brown: nâu, orange: cam, yellow: vàng) |
| 17 | ring\_number | Số lượng vòng (one: một, two: hai, none: không có) |
| 18 | ring\_type | Kiểu vòng (pendant: bền bỉ, evanescent: chóng phai, large: lớn, flaring: hoa hồng, none: không) |
| 19 | spore\_print\_color | Màu bào tử (buff: màu da, pink: hồng, white: trắng, brown: nâu, chocolate: màu socola, gray: xám tro, purple: tím, black: đen, red: đỏ, yellow: vàng, orange: cam, green: xanh lá) |
| 20 | population | Mật độ (several: riêng biệt, cá nhân; solitary: cô độc;  Scattered: rải rác, abundant: nhiều, numerous: rất nhiều, clustered: nhóm thành cụm) |
| 21 | habitat | Môi trường sống (woods: thân gỗ, grasses: bãi cỏ, paths: lối đi, leaves: lá, urban:đô thị, meadows: đồng cỏ, waste: đất bỏ hoang) |
| 22 | mushroom | Phân loại nấm (edible: ăn được, poisonous: độc) |

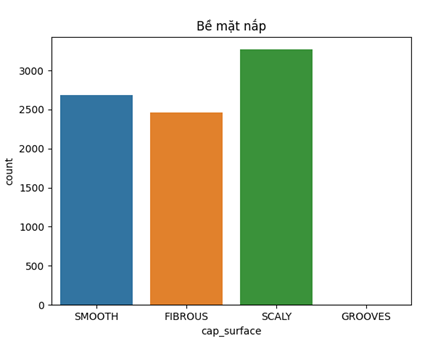
### **2.2.2 Xác định bài toán học máy**

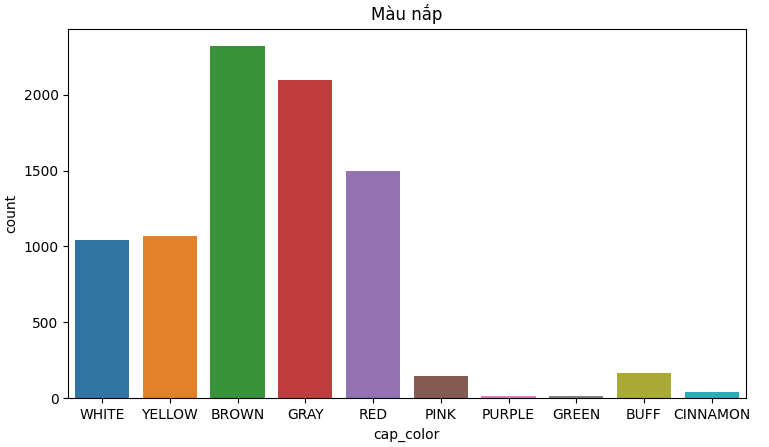
Bài toán phân lớp nhị phân dựa vào đặc tính của loài nấm để phân lớp nấm ăn được hay có độc.

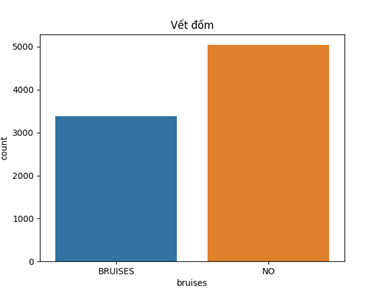
Kỹ thuật sử dụng: Logistic Regression

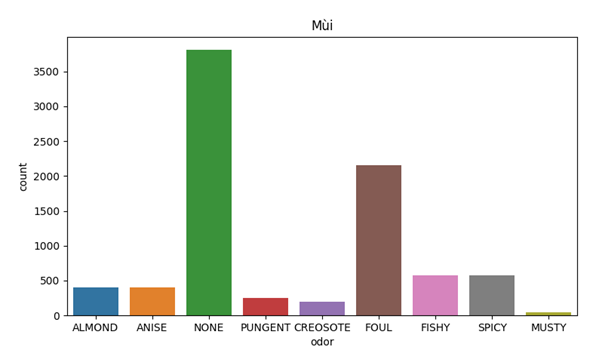
Đầu vào bài toán: 21 thuộc tính đầu tiên mô tả về đặc tính của loài nấm như ở bảng trên(cap\_shape, cap\_surface, cap\_color, bruises, odor, gill\_attachment, gill\_spacing, gill\_size, gill\_color, stalk\_shape, stalk\_root, stalk\_surface\_above\_ring, stalk\_surface\_below\_ring, stalk\_color\_below\_ring, veil\_type, veil\_type, veil\_color, ring\_number, ring\_type, spore\_print\_color, population, habitat). Dưới đây là hình biểu diễn giá trị của một vài thuộc tính tiêu biểu.



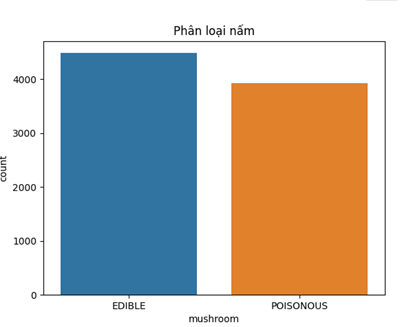








## **-** Đầu ra: nhãn ‘mushroom’ phân loại ăn được và độc.



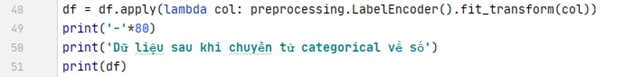
### **2.2.3. Tiền xử lý dữ liệu**

- Xóa dữ liệu NAN

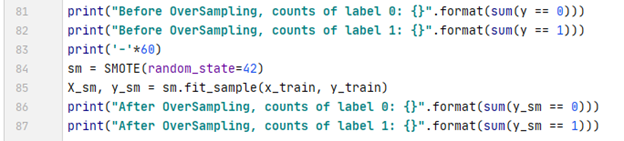


- Xóa cột stalk\_root do có nhiều giá trị bị thiếu.

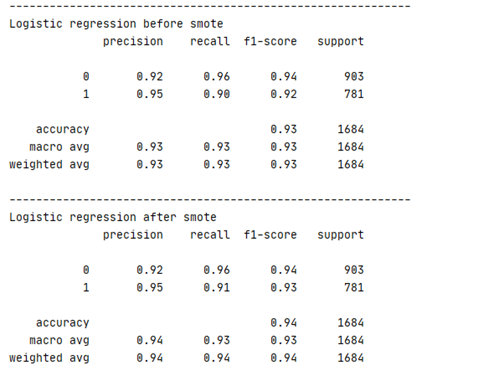


- Dùng LabelEncoder để mã hóa nhãn dữ liệu từ categorical về dạng số

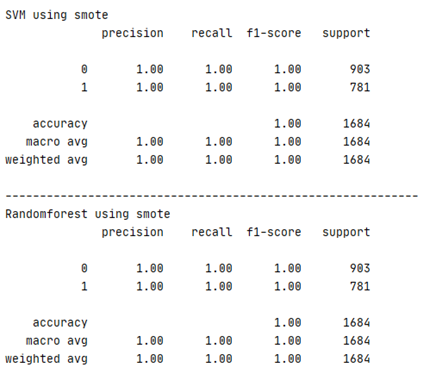
- Dùng smote để xử lý mất cân bằng dữ liệu (do dữ liệu khá đẹp, gần như cân bằng, nên việc xử lý mất cân bằng với bộ dữ liệu này không có ảnh hưởng nhiều đến kết quả đầu ra)

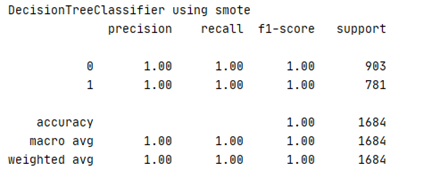


- Kết quả chương trình sau khi sử dụng kỹ thuật logistic regression



- So sánh kết quả khi chạy với một số kỹ thuật phân lớp khác

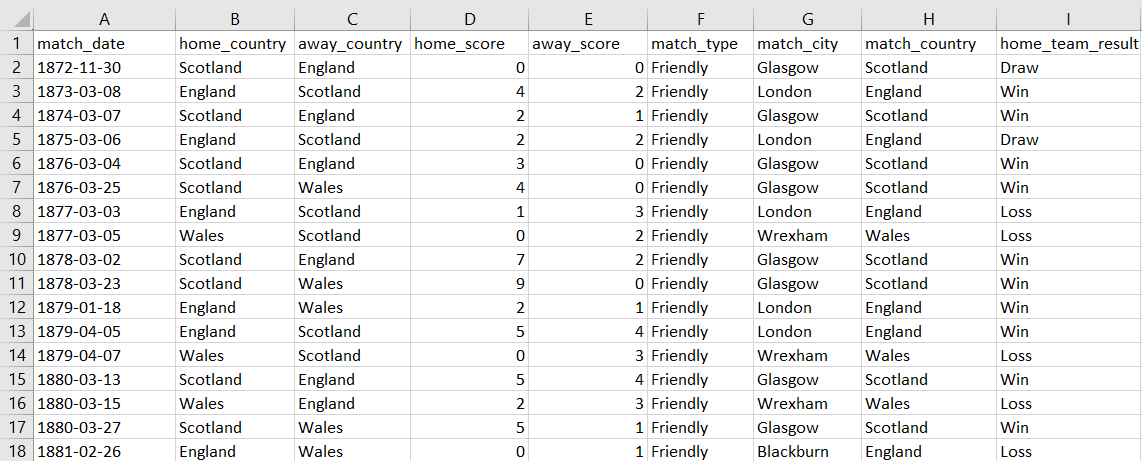




## **2.3. Bài toán phân lớp kết quả (thắng, thua, hòa) trận đá bóng**

### **2.3.1. Mô tả bộ dữ liệu**

- Bộ dữ liệu soccer\_international\_history.csv gồm 9 thuộc tính (9 cột) và 38686 dòng dữ liệu:



- Link tải: https://www.mldata.io/dataset-details/soccer\_international\_history/?fbclid=IwAR0vpWNgJktDZbAnsU2Gm3TWLmEaQ90Hxt5wODF1pxTsxR3oSzAxGo5Pfmc

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Mô tả** |
| 1 | match\_date | Ngày, tháng, năm thi đấu |
| 2 | home\_country | Tên nước chủ nhà |
| 3 | away\_country | Tên nước đội khách |
| 4 | home\_score | Sô bàn thắng của đội nhà |
| 5 | away\_score | Số bàn thắng đội khách |
| 6 | match\_type | Loại thi đấu: Friendly (Giao hữu) |
| 7 | match\_city | Tên thành phố (nơi thi đấu) |
| 8 | match\_country | Tên nước để đối chiếu với kết quả thi đấu |
| 9 | home\_team\_result | Kết quả thi đấu: Win (thắng), Draw (hòa), Loss (thua) |

### 

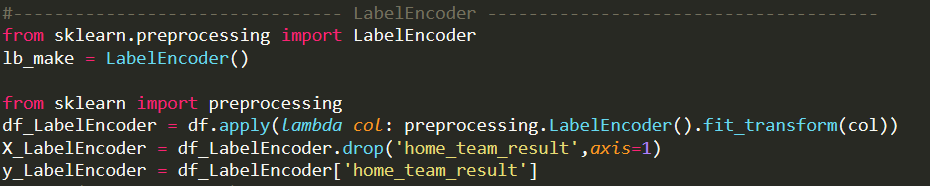
### **2.3.2. Xác định bài toán học máy**

- Với bộ dữ liệu này, ta sử dụng bài toán phân lớp nhiều lớp (3 lớp)

* Đầu vào X là 8 thuộc tính (cột) đầu tiên: match\_date, home\_country, away\_country, home\_score, away\_score, match\_type, match\_city, match\_country
* Đầu ra y là thuộc tính (cột) cuối cùng: home\_team\_result
* Áp dụng bài toán phân lớp sẽ y chia thành 3 lớp: Win, Draw, Loss

- Xử lý dữ liệu: Vì dữ liệu chủ yếu là dạng text, ta sử dụng các kỹ thuật tiền sử lý dữ liệu:

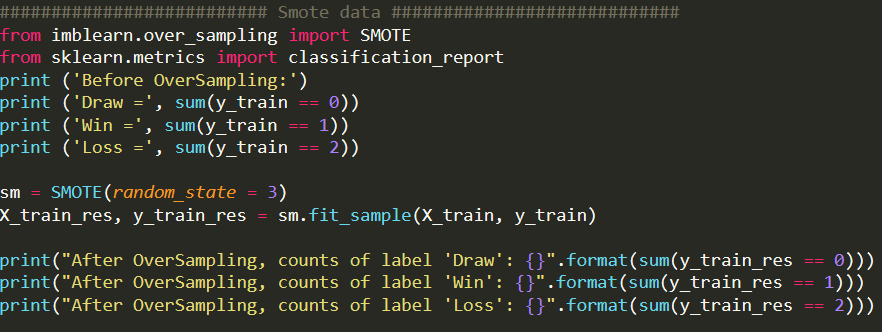
* LabelEncoder để mã hóa về số để tính toán



* Chia tập train và test:



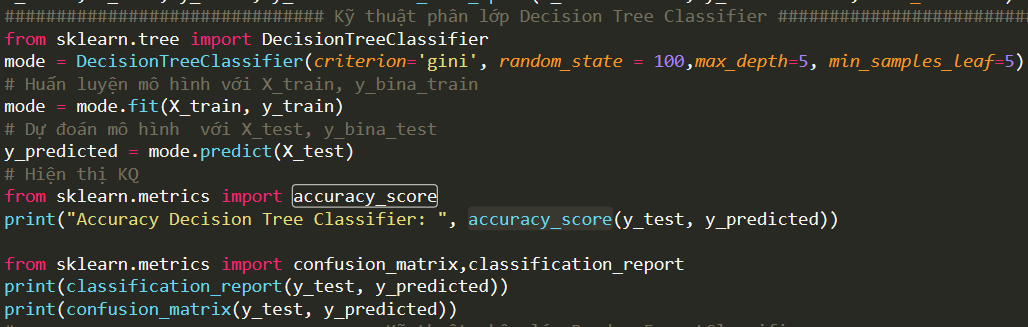
+ SMOTE để cân bằng dữ liệu:



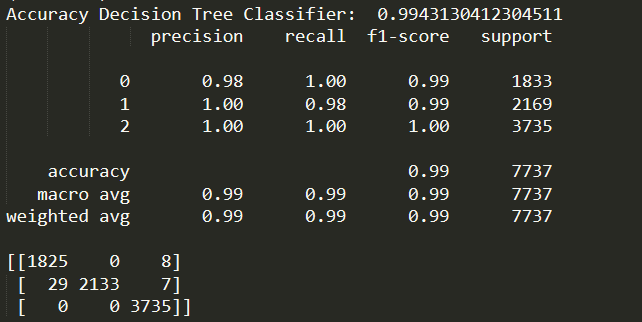
- Các kỹ thuật học máy được sử dụng: Decision Tree Classifier, Random Forest Classifier

### **2.3.3. Phương pháp đánh giá của các kỹ thuật máy**

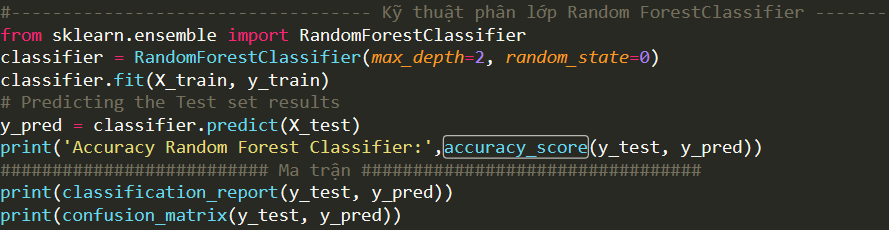
- Kỹ thuật học máy Decision Tree Classifier sử dụng phương pháp đánh giá: accuracy\_score, confusion\_matrix, classification\_report



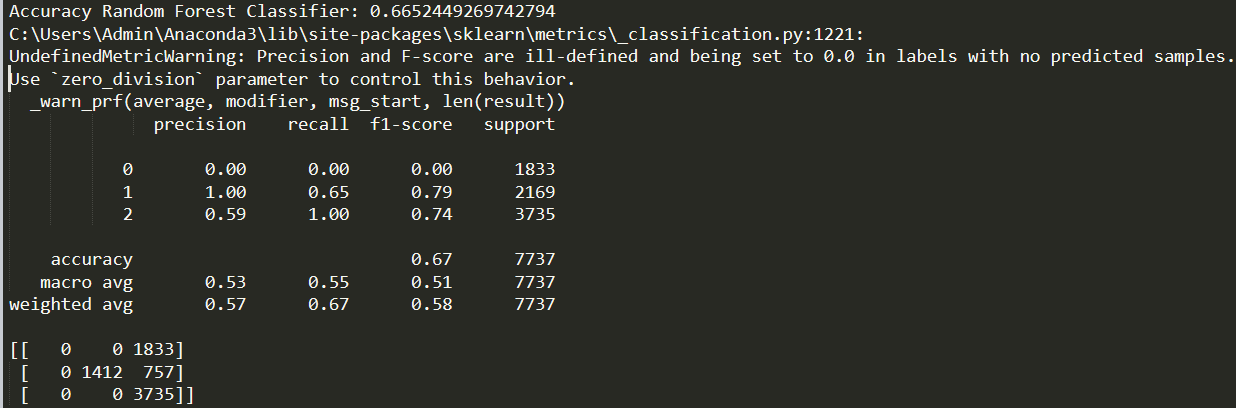
Kết quả chạy:



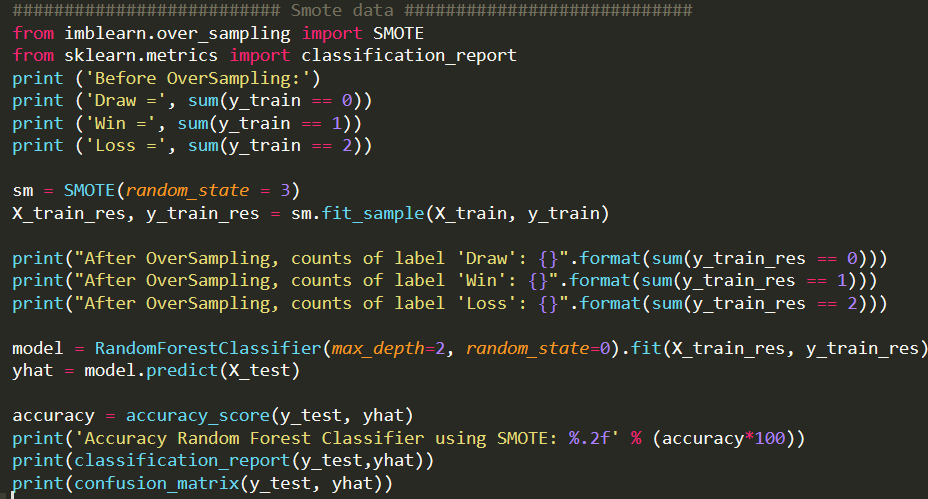
* Kỹ thuật Random Forest Classifier sử dụng phương pháp đánh giá: accuracy\_score, confusion\_matrix, classification\_report



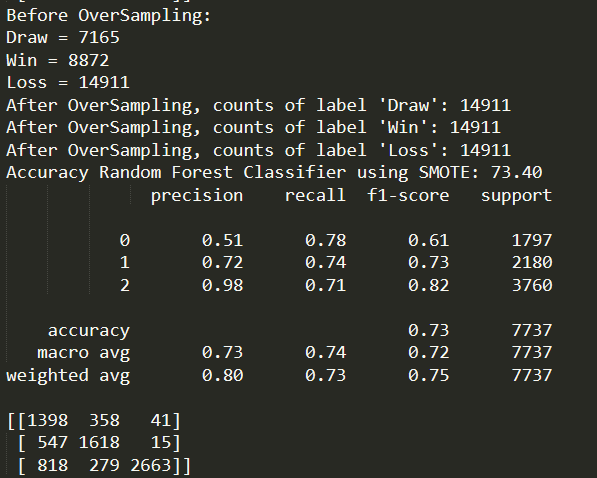
Kết quả chạy



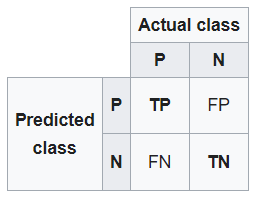
- Độ chính xác của kỹ thuật Random Forest Classifier tăng lên sau khi SMOTE (cân bằng dữ liệu):



Kết quả chạy:



* Phương pháp đáng giá confusion\_matrix và một số công thức tính:



Trong đó:

P = Positive;

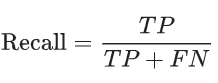
N = Negative;

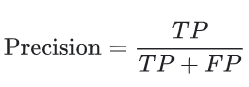
TP = True Positive;

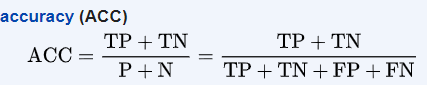
FP = False Positive;

TN = True Negative;

FN = False Negative.







### **2.3.4. Kết luận**

- Kỹ thuật phân lớp Random Forest Classifier nhạy cảm với dữ liệu không cân bằng

# CHƯƠNG III. KẾT LUẬN

3.1. Ứng dụng Yolo

* Sau thời gian tìm hiểu chúng em đã

# 

# **KẾT LUẬN**

Trong quá trình nghiên cứu, tìm hiểu và hoàn thành đề tài báo cáo “Kiến trúc mạng Darknet và ứng dụng YOLO”, chúng em đã thu nhận được thêm những kiến thức và em cũng nhận thấy phân cụm dữ liệu là một lĩnh vực nghiên cứu rộng lớn, còn nhiều điều mà chúng ta cần khám phá. Trong báo cáo này, chúng em đã cố gắng tập trung tìm hiểu và nghiên cứu tổng quan về kiến trúc mạng Darknet. Và cài đặt thử nghiệm kiến trúc mạng Darknet vào ứng dụng YOLO để phát hiện các đối tượng trong cả hình ảnh.

Kết quả đạt được:

* Tìm hiểu về kiến trúc mạng Darknet, Opencv, Python
* Áp dụng xây dựng tìm hiểu ứng dụng YOLO
* Tích lũy thêm nền tảng kiến thức về học máy

Do thời gian tìm hiểu và thực hiện xây dựng ứng dụng còn ít nên không trách khỏi thiếu sót. Trong thời gian tới, chúng em sẽ cố gắng tiếp tục nghiên cứu, tìm hiểu thêm một số kiến trúc mạng khác và nhất là có thể tìm hiểu và phát triển các kỹ thuật phân đoạn ảnh để có thể xử lý với ảnh động, video. Rất mong nhận được sự góp ý của giảng viên và các bạn.

Chúng em xin chân thành cảm ơn!

# **TÀI LIỆU THAM KHẢO**

1. [Deep Learning](https://www.cs.toronto.edu/~hinton/absps/NatureDeepReview.pdf) – Yann L., Yoshua B. & Geoffrey H. (2015)

2. [TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems](http://download.tensorflow.org/paper/whitepaper2015.pdf) – Martín A., Ashish A. B., Eugene B. C., et al. (2015)

3. [TensorFlow: a system for large-scale machine learning](https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf%20rel=) – Martín A., Paul B., Jianmin C., Zhifeng C., Andy D. et al. (2016)

4. [Deep learning in neural networks](https://arxiv.org/pdf/1404.7828.pdf) – Juergen Schmidhuber (2015)

5. [Human-level control through deep reinforcement learning –](https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassibis15NatureControlDeepRL.pdf) Volodymyr M., Koray K., David S., Andrei A. R., Joel V et al (2015)

6. [Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks](https://arxiv.org/pdf/1506.01497.pdf) – Shaoqing R., Kaiming H., Ross B. G. & Jian S. (2015)

7. [Long-term recurrent convolutional networks for visual recognition and description](https://arxiv.org/pdf/1411.4389.pdf) – Jeff D., Lisa Anne H., Sergio G., Marcus R., Subhashini V. et al. (2015)

8.Yolov5: https://github.com/ultralytics/yolov5