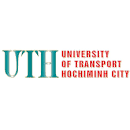
|  |
| --- |
| **TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI TP. HỒ CHÍ MINH**  **KHOA ĐIỆN – ĐIỆN TỬ VIỄN THÔNG** |



**LUẬN VĂN TỐT NGHIỆP**

**NHẬN DẠNG CỬ CHỈ ĐỘNG CỦA TAY NGƯỜI**

**BẰNG TRÍ TUỆ NHÂN TẠO**

Ngành: **ĐIỆN**

Chuyên ngành: **ĐIỆN TỬ VIỄN THÔNG**

Giảng viên hướng dẫn : TS. Nguyễn Quang Sang

Sinh viên thực hiện : Trần Văn Hải Triều

MSSV: 1851040065 Lớp: DV18

TP. Hồ Chí Minh, 2023

**CAM KẾT LIÊM CHÍNH HỌC THUẬT**

Tôi cam kết rằng kết quả nghiên cứu được trình bày trong quyển luận văn này là kết quả học và lao động của tôi dưới sự hướng dẫn của giảng viên hướng dẫn. Kết quả nghiên cứu trong luận văn là trung thực và chưa từng được công bố trong bất cứ công trình nghiên cứu nào khác. Các kết quả nghiên cứu và tài liệu trong luận văn này đã được liệt kê nguồn và ghi trích dẫn cụ thể.

Tôi hoàn toàn chịu trách nhiệm trước Hội đồng bảo vệ luận văn, Trường đại học Giao Thông Vận Tải và pháp luật về những cam kết nêu trên.

Thành phố Hồ Chí Minh, ngày… tháng … năm

Tác giả luận văn

**LỜI CẢM ƠN**

Xin trân trọng cảm ơn đến các quý Thầy Cô khoa Điện – Điện Tử Viễn Thông, Trường Đại học Giao Thông Vận Tải TP. Hồ Chí Minh đã tạo điều kiện, hỗ trợ để luận văn này được hoàn tất. Từ những kiến thức được học và sự gợi ý từ các thầy cô chính là cơ sở cho đề tài luận văn này.

Xin gửi lời cảm ơn đặc biệt nhất đến thầy Nguyễn Quang Sang đã gợi ý đề tài nghiên cứu, cũng như nhiệt tình hỗ trợ, góp ý trong quá trình thực hiện luận văn để nội dung được hoàn thiện.

Nội dung của luận văn là kết quả của quá trình tự nghiên cứu cho nên những hạn chế, thiếu sót là không thể tránh khỏi. Do đó hi vọng các Thầy Cô góp ý và chia sẻ để nội dung của luận văn sẽ hoàn chỉnh và phát triển thêm nữa.

**MỤC LỤC**

[CHƯƠNG 1: GIỚI THIỆU VỀ TRÍ TUỆ NHÂN TẠO 1](#_Toc122282284)

[1. Trí tuệ nhân tạo (Artificial Intelligence) 1](#_Toc122282285)

[2. Machine learning 2](#_Toc122282286)

[3. Deep learning 3](#_Toc122282287)

[CHƯƠNG 2: VECTOR, MA TRẬN VÀ LẬP TRÌNH PYTHON 4](#_Toc122282288)

[1. Vector và không gian 4](#_Toc122282289)

[1.1. Định nghĩa không gian vector 4](#_Toc122282290)

[1.2. Không gian vector Rn 5](#_Toc122282291)

[1.3. Sự độc lập tuyến tính – sự phụ thuộc tuyến tính 5](#_Toc122282292)

[1.3.1. Tổ hợp tuyến tính 5](#_Toc122282293)

[1.3.2. Độc lập tuyến tính và phụ thuộc tuyến tính 6](#_Toc122282294)

[1.4. Không gian con 6](#_Toc122282295)

[1.5. Cơ sở 6](#_Toc122282296)

[1.6. Tích vô hướng và tích vector 7](#_Toc122282297)

[1.6.1. Tích vô hướng 7](#_Toc122282298)

[1.6.2. Tích vector 7](#_Toc122282299)

[2. Ma trận 7](#_Toc122282300)

[2.1. Định nghĩa 7](#_Toc122282301)

[2.2. Các phép toán ma trận 7](#_Toc122282302)

[3. Cơ bản về cơ sở dữ liệu (database) 8](#_Toc122282303)

[3.1. Cơ sở dữ liệu quan hệ và phi quan hệ 8](#_Toc122282304)

[3.2. SQL và joins 9](#_Toc122282305)

[3.3. NoSQL 13](#_Toc122282306)

[4. Dữ liệu dạng bảng 14](#_Toc122282307)

[5. Giới thiệu ngôn ngữ python 14](#_Toc122282308)

[5.1. Khái niệm 14](#_Toc122282309)

[5.2. Python có thể làm gì? 15](#_Toc122282310)

[5.3. Tại sao lại chọn Python? 16](#_Toc122282311)

[6. Cơ bản về lập trình Python 16](#_Toc122282312)

[6.1. Biểu thức (Expressions) trong Python 16](#_Toc122282313)

[6.2. Cấu trúc dữ liệu (Data Structure) 24](#_Toc122282314)

[6.2.1. List 24](#_Toc122282315)

[6.2.2. Dictionary 28](#_Toc122282316)

[6.3. Hàm (Function) 31](#_Toc122282317)

[CHƯƠNG 3: MÁY HỌC – MACHINE LEARNING 37](#_Toc122282318)

[1. Khái niệm 37](#_Toc122282319)

[1.1. Machine learning 37](#_Toc122282320)

[1.2. Phân loại 38](#_Toc122282321)

[1.3. Ứng dụng 39](#_Toc122282322)

[2. Thư viện Numpy 40](#_Toc122282323)

[3. Dữ liệu và xử lí dữ liệu 47](#_Toc122282324)

[4. Học giám sát (Supervised Learning) 54](#_Toc122282325)

[4.1. Hồi quy tuyến tính (Linear Regression) 54](#_Toc122282326)

[4.2. Logistic Regression 64](#_Toc122282327)

[5. Học không giám sát (Unsupervised learning) 70](#_Toc122282328)

[5.1. Phân cụm K-means 70](#_Toc122282329)

[CHƯƠNG 4: MẠNG THẦN KINH NHÂN TẠO – ATIFICIAL NEURAL NETWORKS 76](#_Toc122282330)

[1. Giới thiệu 76](#_Toc122282331)

[2. Mạng neuron tích chập – Convolution Neural Network 83](#_Toc122282332)

[3. Bài toán nhận diện cử chỉ 88](#_Toc122282333)

[3.1. Opencv 88](#_Toc122282334)

[3.2. Tensorflow và Keras 88](#_Toc122282335)

[3.2.1. Các khái niệm trong Keras 88](#_Toc122282336)

[3.2.2. Xây dựng bài toán nhận diện cử chỉ 89](#_Toc122282337)

[CHƯƠNG 5: KẾT LUẬN 99](#_Toc122282338)

[1. Một số điểm hạn chế của luận văn 100](#_Toc122282339)

[2. Hướng phát triển của luận văn 100](#_Toc122282340)

[3. Kết luận 100](#_Toc122282341)

**DANH MỤC HÌNH ẢNH**

[Hình 1 – AI, Machine learning và Deep learing 1](#_Toc122285397)

[Hình 2 - Ứng dụng của Machine learning 2](#_Toc122285398)

[Hình 3 – Khác nhau giữa Machine learning và Deep learning 3](#_Toc122285399)

[Hình 4 – Dữ liệu dạng bảng 14](#_Toc122285400)

[Hình 5 – Ngôn ngữ lập trình được ưa chuộng trong năm 2022 15](#_Toc122285401)

[Hình 6 – Các ứng dụng của Python 16](#_Toc122285402)

[Hình 7 – Các đặc điểm của Python 16](#_Toc122285403)

[Hình 8 – Ứng dụng của machine learning 39](#_Toc122285404)

[Hình 9 - Ứng dụng của machine learning trong y học 40](#_Toc122285405)

[Hình 10 - Ứng dụng chặn spam 40](#_Toc122285406)

[Hình 11 - Quá trình thay đổi của dữ liệu 47](#_Toc122285407)

[Hình 12 – Neuron sinh học 77](#_Toc122285408)

[Hình 13 – Cấu trúc mạng neuron đơn giản 77](#_Toc122285409)

[Hình 14 – Mạng chuyển tiếp một lớp 78](#_Toc122285410)

[Hình 15 – Mạng chuyển tiếp nhiều lớp 79](#_Toc122285411)

[Hình 16 – Mạng một nút phản hồi 79](#_Toc122285412)

[Hình 17 – Mạng đệ qui một lớp 80](#_Toc122285413)

[Hình 18 – Mạng đệ qui đa lớp 81](#_Toc122285414)

[Hình 19 – Mạng tích chập với mạng neuron thông thường 83](#_Toc122285415)

[Hình 20 – Cấu trúc của một dữ liệu ảnh 84](#_Toc122285416)

[Hình 21 – Minh họa tích chập 84](#_Toc122285417)

[Hình 22 - Max pooling 86](#_Toc122285418)

[Hình 23 – Minh họa một mô hình đầy đủ 86](#_Toc122285419)

[Hình 24 – VGG16 87](#_Toc122285420)

[Hình 25 - Inception 87](#_Toc122285421)

[Hình 26 - ResNet 87](#_Toc122285422)

[Hình 27 – Darknet - 53 88](#_Toc122285423)

**DANH MỤC BẢNG**

[Bảng 1 – Các toán tử trong Python 18](#_Toc122285424)

[Bảng 2– Toán tử logic 20](#_Toc122285425)

[Bảng 3 – Mức độ ưu tiên của các toán tử 22](#_Toc122285426)

[Bảng 4 - Các hàm trong thư viện math 23](#_Toc122285427)

**TÓM TẮT**

Nội dung của luận văn sẽ xoay quanh chủ yếu về các biểu thức toán học cũng như là sử dụng ngôn ngữ lập trình python để triển khai những thuật toán đó đưa vào các mô hình của machine learning, bên cạnh đó cũng giới thiệu và mô phỏng một dự án deep learning đơn giản để minh họa rõ hơn cách máy áp dụng các công thức toán vào trong một môi trường thực tế. Tất cả sẽ được giới thiệu từ lý thuyết toán học nền tảng cho đến một số các kiến thức nâng cao, cụ thể như sau:

.Giới thiệu sơ lược về khái niệm trí tuệ nhân tạo, máy học và học sâu, sự khác nhau cơ bản giữa chúng

. Giới thiệt ngôn ngữ Python và các cấu trúc dữ liệu.

. Chi tiết về machine learning và các thuật toán tiêu biểu.

. Chi tiết deep learning và tổng quan về một mô hình deep learning (nhận dạng cử chỉ tay: máy sẽ trả về kí tự tương ứng với cử chỉ tay được đưa trước camera).

**LÝ DO CHỌN ĐỀ TÀI**

Trí tuệ nhân tạo là một lĩnh vực rất nổi tiếng, thịnh hành đã và đang phát triển trong nhiều năm qua, chúng ta không hiếm để bắt gặp những ứng dụng của nó trong y tế giáo, giáo dục, tài chính, ngân hàng, …Một trong số những ứng dụng phổ biến đó là bài toán nhận diện như nhận diện khuôn mặt, nhận diện đeo khẩu trang, nhận diện số người,… Bên cạnh đó, cũng có những bài toán nhận diện cử động của con người tiêu biểu là bài toán nhận diện cử chỉ động của tay người như đề tài này. Nhận diện cử chỉ tay không phải là một phạm trù mới, đã được sử dụng rất nhiều như điều khiển các máy móc vận hành bằng cử chỉ tay, điều khiển các thiết bị ngoại vi như màn hình TV. Vậy nên, lí do để đề tài được triển khai gồm các ý sau:

Thứ nhất: Việc áp dụng trí tuệ nhân tạo là một lĩnh vực rất phổ biến, được rất nhiều người quan tâm. Chính khả năng vận hành thông minh đã tạo cảm hứng, động lực để em nghiên cứu và tìm hiểu.

Thứ hai: Bài toán nhận diện cử chỉ động của tay em xây dựng không đơn thuần là xác nhận sự hiện diện của bàn tay, mà từ chính hình dạng, cử chỉ của bàn tay máy sẽ nhận diện ra một từ, một chữ có thể dành để giao tiếp cho người khiếm khuyết không thể nghe và nói hoặc thực hiện một tác vụ nào đó.

Thứ ba: Nhờ vào bài toán này em có thể biết thêm về việc làm thế nào để một mô hình trí tuệ nhân tạo có khả năng nhận diện, cách mà nó được xây dựng cũng như quá trình thực hiện.

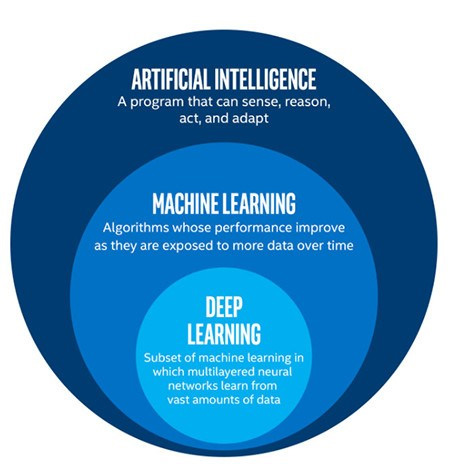
Tất cả ý trên chính là điều kiện để đề tài này được lựa chọn để triển khai nghiên cứu và trình bày.

1. GIỚI THIỆU VỀ TRÍ TUỆ NHÂN TẠO
   1. Trí tuệ nhân tạo (****Artificial Intelligence)****

Trí tuệ nhân tạo về cơ bản là cơ chế kết hợp trí thông minh của con người vào máy móc thông qua một bộ qui tắc (thuật toán). AI là sự kết hợp của hai từ: ‘Artificial’ nghĩa là thứ được tạo ra bởi con người hoặc là thứ mà không có trong tự nhiên và ‘Intelligence’ nghĩa là khả năng hiểu hoặc tư duy tương ứng. Hoặc ta có một định nghĩa khác là: ‘AI về cơ bản là việc nghiên cứu đào tạo máy móc (máy tính) bắt chước bộ não và khả năng tư duy của con người’. AI tập trung vào ba khía cạnh chính là học hỏi, suy luận và tự điều chỉnh để đạt được hiệu quả tốt nhất.

Có một sự thật là chúng ta không sao chép hoàn toàn bộ não và khả năng tư duy của con người. Một ví dụ điển hình cho AI ngày nay là Google Assistant. Khi ta đưa ra một câu truy vấn dạng tín hiệu giọng nói của sẽ biến đổi thành tín hiệu số hoặc bất kì dạng tín hiệu nào mà máy tính có thể hiểu được, kế đến dữ liệu đã biến đổi sẽ được đưa vào bộ phận xử lí và trả về kết quả với dạng thông tin tương ứng, cuối cùng chuyển đổi ngược lại thành tín hiệu giọng nói để phát qua loa của thiết bị.

Việc nghiên cứu AI là một vấn đề rất phức tạp bởi vì để mô phỏng hoàn toàn bộ não con người là điều rất khó. Chúng ta vẫn chưa khai phá hết được khía cạnh của bộ não con người cho đến bây giờ chẳng hạn như tại sao chúng ta lại nằm mơ,.v..v. Bên cạnh đó AI là một khái niệm rộng hơn với Machine learning và Deep learning nằm trong nó.

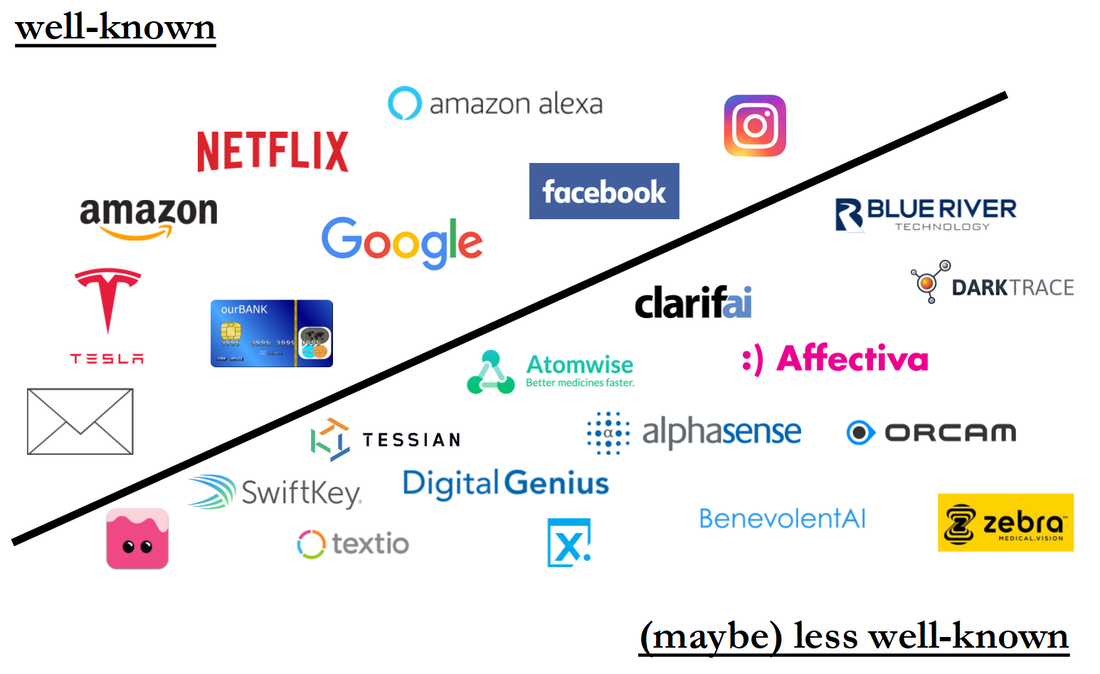


Hình 1 – AI, Machine learning và Deep learing

* 1. Machine learning

Machine learning là việc nghiên cứu hay xử lí cung cấp cho máy tính khả năng học một cách tự động trên trải nghiệm mà nó có được và khả năng tự cải thiện mà không cần lập trình một cách cụ thể. Machine learning là một bộ con của AI. Machine learning tập trung vào việc phát triển những chương trình để nó có thể truy nhập dữ liệu và tự sử dụng chúng. Toàn bộ quá trình khám phá dữ liệu để xác định những mẫu có thể đang được hình thành và đưa ra quyết định tương lai tốt hơn theo các thông tin cung cấp cho nó. Mục đích chính của Machine learning là cho phép hệ thống tự học thông qua trải nghiệm mà không có bất kì sự can thiệp hay hỗ trợ của con người.

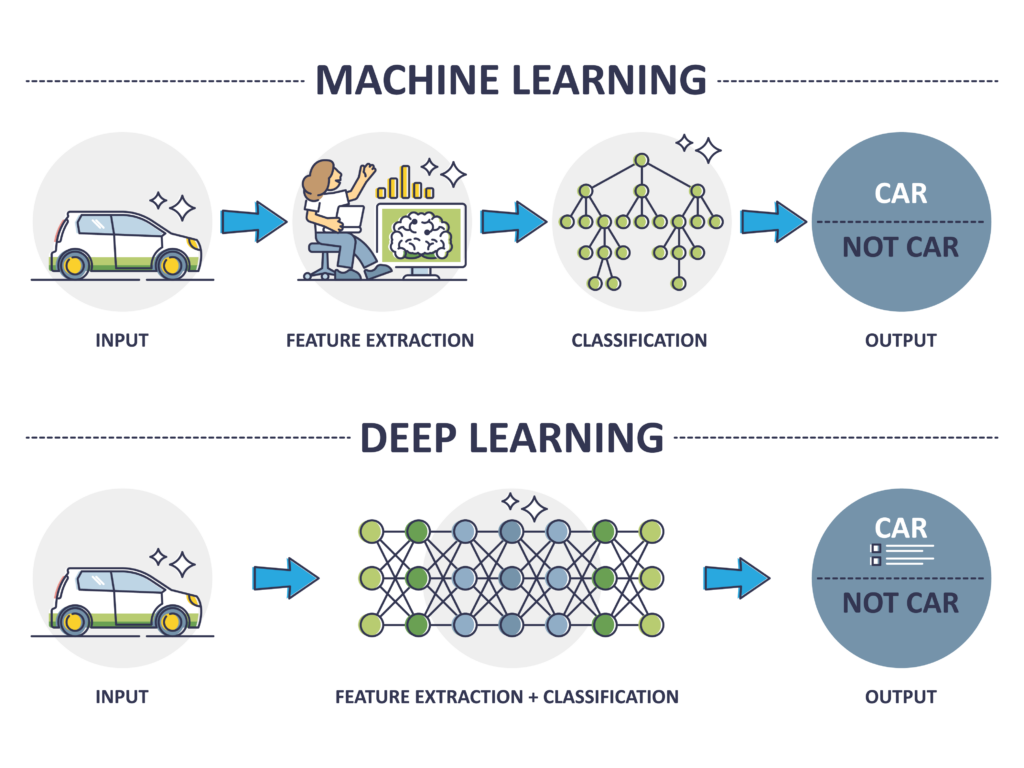
Một vài ví dụ của Machine learning là các hệ thống đề xuất như Google hay Netflix để đưa ra các kết quả liên quan hay những kết quả mà khách hàng có thể quan tâm.



Hình 2 - Ứng dụng của Machine learning

* 1. Deep learning

Deep learning là một phần của Machine learning , sử dụng mạng thần kinh (Neural Network) để bắt chước hành vi giống não con người. Những thuật toán Deep learning tập trung vào cơ chế mẫu xử lí thông tin để có thể xác định các mẫu giống như não con người của chúng ta và phân loại thông tin theo đó. Deep learning làm việc trên một bộ dữ liệu lớn hơn so với Machine learning và có khả năng tự quản lí cơ chế dự đoán.



Hình 3 – Khác nhau giữa Machine learning và Deep learning

Deep learning thực chất là những thuật toán Machine learning sử dụng mạng thần kinh sâu (nhiều hơn một lớp) để phân tích dữ liệu và cung cấp kết quả tương ứng. Lấy ví dụ ta có một đèn flash và ta muốn dạy cho mô hình Machine learning biết rằng khi ai đó nói ‘dark’ thì đèn sẽ bật, mô hình sẽ phân tích các cụm từ khác nhau và sẽ tìm kiếm từ ‘tối’ để bật bèn nhưng khi ta nói một câu phức tạp hơn ‘Tôi không thấy gì cả, đèn mờ quá’ thì đèn sẽ không bật. Nếu là mô hình Deep learning thì đèn sẽ bật, một mô hình deep learning có thể học từ phương pháp tính toán của chính nó.

1. VECTOR, MA TRẬN VÀ LẬP TRÌNH PYTHON
   1. 1. Vector và không gian
      1. Định nghĩa không gian vector

Vector bao gồm hai phần chủ yếu là độ lớn và hướng hay như khái niệm phổ thông thì vector là một đoạn thẳng có hướng. Người ta thường dùng điểm đầu và điểm cuối để thể hiện vector bằng chữ in hoa, bên cạnh đó có thể sử dụng chữ in thường.

, , .

,, .

Trong đại số tuyến tính, vector được thể hiện trong không gian ( thường là hai chiều) với kí hiệu = (x, y) với x là tọa độ của trục hoành và y là tọa độ của trục tung. Ví dụ:

= (5, 0)

Bên cạnh đó chúng ta cũng có thể ký hiệu = .

**Định nghĩa**: Giả sử V là một tập hợp mà các phần tử được kí hiệu bởi , , ,..., R là một trường số. Trên V có một phép toán gọi là phép cộng hai phần tử của V (kí hiệu "+") và phép toán thứ hai gọi là phép nhân một phần tử của V với một số thuộc trường R (kí hiệu "."). Tập hợp V cùng với hai phép toán này được gọi là một không gian vectơ trên trường R (hay một R-không gian vectơ) nên các điều kiện sau được thoả mãn đối với mọi , , ,.. ∈ V và mọi r, s, 1,.. ∈ R.

1) (+ ) + = + ( + );

2) + = + ;

3) có một phần tử ∈ V thoả mãn điều kiện: + = ;

4) với mỗi α ∈ V có một phần tử, kí hiệu bởi - α , cũng thuộc V thoả mãn điều kiện: + (- ) = ;

5) r( + ) = r + r ;

6) (r + s) = r + s ;

7) (rs) = r (s ) ;

8) 1. = .

∈ V được gọi là một vectơ, được gọi là vectơ không, - được gọi là vectơ đối của .

* + 1. Không gian vector Rn

Gọi Rn  là tập hợp mà mỗi phần tử là một bộ n số thực có thứ tự (1, 2,...,n). Xét 2 phần tử , ∈ Rn.

= (1, 2,...,n); = (1, 2,...,n).

Ta định nghĩa phép cộng và phép nhân như sau:

( + ) = (1 + 1, 2 + 2 ,…, n + n).

k = (k1, k2,…, kn), k ∈ R.

Rn là một không gian vector.

* + 1. Sự độc lập tuyến tính – sự phụ thuộc tuyến tính
       1. Tổ hợp tuyến tính

V là một không gian vector, S = {1, 2,...,n} là một hệ gồm n vector của V.

= c11 + c22 + ... + cnn, ci ∈ R (i=1, 2,.., n) gọi là một tổ hợp tuyến tính của hệ S.

Định nghĩa: Nếu là tổ hợp tuyến tính của hệ gồm n vector 1, 2,...,n thì ta nói biểu diễn tuyến tính (hay biểu thị tuyến tính) được qua hệ vector 1, 2,...,n.

* + - 1. Độc lập tuyến tính và phụ thuộc tuyến tính

Cho V là một không gian vector và S = {1, 2,...,n} là một hệ các vector trong V.

Định nghĩa:

. Nếu đẳng thức c11 + c22 + ... + cnn = chỉ xảy ra khi c1=c2=…=cn=0 thì ta nói hệ S độc lập tuyến tính.

. Nếu đẳng thức c11 + c22 + ... + cnn = xảy ra với ít nhất một hệ số ci ≠ 0 thì ta nói hệ S phụ thuộc tuyến tính.

Tóm lại để xét độc lập tuyến tính hay phụ thuộc tuyến tính của hệ S = {ui}. Ta xét đẳng thức iui = 0, đẳng thức này tương đương một hệ phương trình tuyến tính thuần nhất với các ẩn là I. Nếu hệ có nghiệm không tầm thường thì hệ S phụ thuộc.

* + 1. Không gian con

Cho không gian vector V cùng hai phép toán cộng và nhân. W là tập con của V và W khác rỗng. Nếu W cùng với hai phép toán (cộng và nhân) thừa hưởng từ V cũng lập thành không gian vector thì ta nói W là không gian vector con (hay gọi tắt là không gian con) của không gian vector V.

* + 1. Cơ sở

Hệ vector B = {1, 2,...,n} gọi là được sắp nếu ta có quan tâm thứ tự của các vector trong hệ, nếu đổi thứ tự của các vector trong hệ thì ta được một hệ vector khác.

Định nghĩa:

Hệ được sắp các vector B = {1, 2,...,n} được gọi là cơ sở của không gian vector V nếu thỏa 3 điều sau:

1. ui  ∈ V, i = .
2. B là độc lập tuyến tính.
3. ∈ V, là tổ hợp tuyến tính của các vector trong hệ B, nghĩa là luôn tìm được cácI ∈ R, i = để cho = c11 + c22 + ... + cnn.
   * 1. Tích vô hướng và tích vector
        1. Tích vô hướng

Kết quả của tích vô hướng bởi hai vector luôn là đại lượng vô hướng. Giả sử hai vector và . Tích vô hướng của hai vector sẽ bằng với độ lớn của , và cosine của gốc hợp bởi hai vector này.

Tích vô hướng = cosα.

* + - 1. Tích vector

Tích vector của hai vector và với gốc α được tính như sau: a x b = sinα

Chú ý rằng tích vector luôn có hướng nhất định và kết quả luôn vuông góc với hai vector thành phần. Trong trường hợp hai vector song song thì kết quả sẽ bằng 0 (sin(0) = 0).

* 1. Ma trận
     1. Định nghĩa

Một ma trận cỡ m x n trên R là một bảng số hình chữ nhật gồm m hàng và n cột có dạng:

A =

Vậy aij ∈ R là phần tử nằm ở hàng i cột j của ma trận A

* + 1. Các phép toán ma trận

Phép chuyển vị ma trận: Nếu A = thì AT = . Thấy ngay nếu A cỡ m x n thì AT có cỡ n x m.

Phép cộng ma trận: A + B =

Phép nhân ma trận với một số: αA =

Phép nhân ma trận với ma trận: A.B = C = (cij)m x n với cij= = [ .

* 1. Cơ bản về cơ sở dữ liệu (database)
     1. Cơ sở dữ liệu quan hệ và phi quan hệ

Cơ sở dữ liệu quan hệ hay được gọi là hệ thống quản lý cơ sở dữ liệu quan hệ (RDBMS) hay SQL databases. Tất cả các cơ sở dữ liệu quan hệ có thể được dùng để mà quản lý những ứng dụng theo định hướng giao dịch (OLTP) và hầu hết các cơ sở dữ liệu phi quan hệ trong các danh mục lưu trữ tài liệu và lưu trữ các cột chữ cũng có thể được dùng cho OLTP điều này làm tăng sự nhầm lẫn giữa hai loại cơ sở dữ liệu.

Cơ sở dữ liệu phi quan hệ hay được gọi là NoSQL databases. NoSQL đã trở thành một định nghĩa tiêu chuẩn của công nghiệp nhưng mà cái tên của nó đang dần mất đi sự phổ biến bởi vì nó không có bao hàm đầy đủ độ phức tạp và phạm vi của nơi lưu trữ cơ sở dữ liệu phi quan hệ đang có.

Dưới đây là sự khác biệt cơ bản của hai cơ sở dữ liệu quan hệ và phi quan hệ:

Cơ sở dữ liệu quan hệ

Ưu điểm:

* Cơ sở dữ liệu quan hệ hoạt động với dữ liệu có cấu trúc (structured data).
* Hỗ trợ tính nhất quán của chuyển giao ACID và hỗ trợ “joins” (phép nối).
* Cơ sở dữ liệu quan hệ kết hợp với tính toàn vẹn của dữ liệu tích hợp và hệ thống và có hệ sinh thái lớn.
* Các mối liên kết trong hệ thống này có tính ràng buộc.
* Có thể lập index vô hạn và SQL mạnh.

Nhược điểm:

* Cơ sử dữ liệu quan hệ không mở rộng theo chiều ngang chỉ theo chiều dọc.
* Dữ liệu được chuẩn hóa, nghĩa là rất nhiều phép nối, điều này ảnh hưởng đến tốc độ.
* Có vấn đề khi hoạt động với các dữ liệu bán cấu trúc.

Cơ sở dữ liệu phi quan hệ/NoSQL

Ưu điểm:

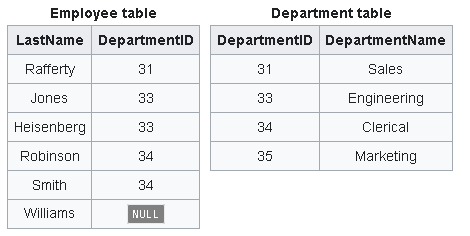
* Có thể mở rộng theo phương ngang và làm việc với dữ liệu bán cấu trúc hay phi cấu trúc. Hỗ trợ một chút với nhất quán chuyển giao ACID.
* Lựa chọn giữa không có giản đồ (Schema-free) hoặc giản đồ trên bản đọc (Schema-on-read).
* Tính khả thi cao.
* Trong khi nhiều cơ sở dữ liệu NoSQL là nguồn mở và “miễn phí”. Hiện nay có nhiều sản phẩm thương mại sẵn có.

Nhược điểm:

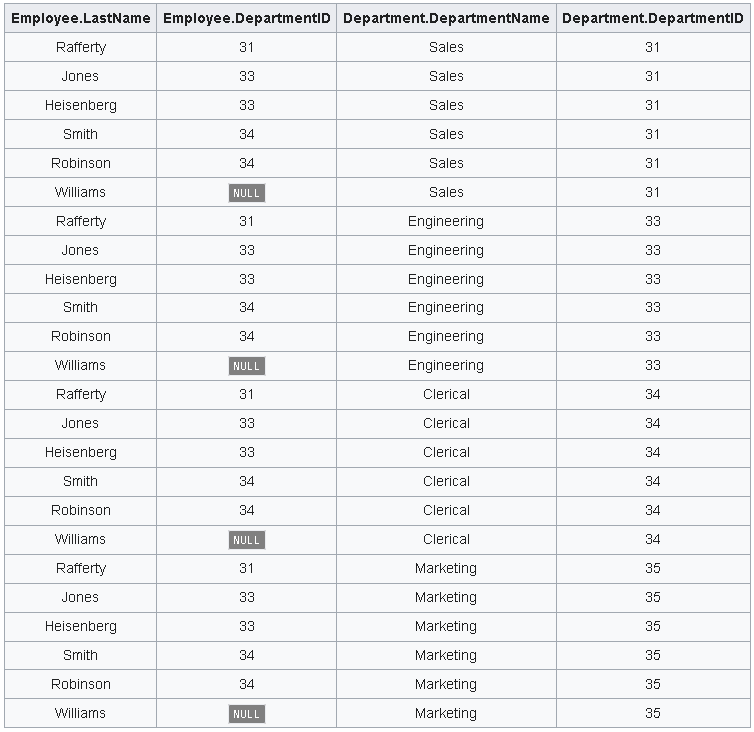
* Tính nhất quán yếu (BASE) thay vì ACID.
* Hỗ trợ cho joins (phép nối) bị giới hạn.
* Dữ liệu không được chuẩn hóa, yêu cầu cập nhập lớn.
* Không có toàn vẹn dữ liệu tích hợp.
* Lập index hữu hạn.
  + 1. SQL và joins

Mệnh đề join trong SQL tương ứng với phép toán hợp trong đại số quan hệ - kết hợp các cột từ một hay nhiều bảng thành một bảng mới. Tiêu chuẩn ANSI qui định 5 loại join là: inner, left outer, right outer, full outer và cross.

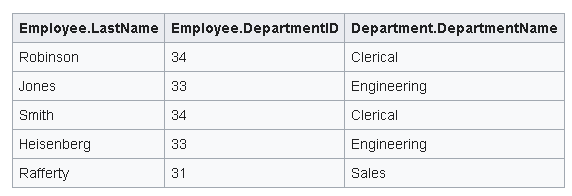
Để giải thích loại join, sử dụng hai bảng sau làm ví dụ:



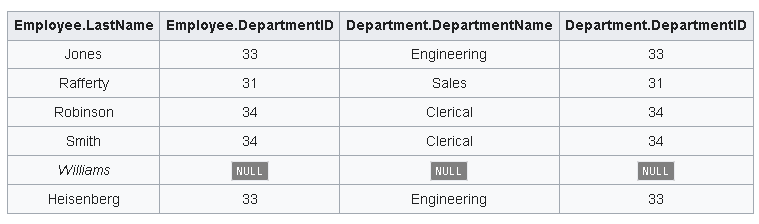
Cross join sẽ tạo ra các hàng là sự kết hợp của mỗi hàng từ bảng thứ nhất với bảng thứ hai.



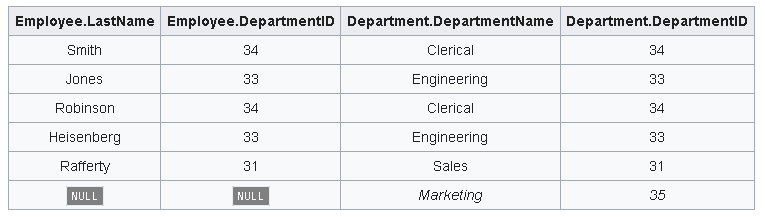
Inner join sẽ tạo ra bảng mà thỏa mãn điều kiện join chẳng hạn như DepartmentID của bảng Employee = DepartmentID của bảng Department.



Left outer join sẽ trả về kết quả luôn chứa những hàng của bảng bên trái (A), thậm chí nếu không thỏa điều kiện join ở bảng bên phải B. Ví dụ dưới đây cho ta cả những tên nhân viên mà chưa được giao tới bộ phận tương ứng khác với inner join ở trên là nhân viên đó không được đính kèm.



Right outer join nó gần giống left outer join, ngoại trừ việc đảo ngược lại.



Full outer join là được áp dụng như sự kết hợp của left outer join và right outer join



* + 1. NoSQL

Cơ sở dữ liệu NoSQL cung cấp một cơ chế cho việc lưu trữ và truy xuất dữ liệu được mô hình hóa bằng các phương tiện khác với những quan hệ dạng bảng trong cơ sở dữ liệu quan hệ.

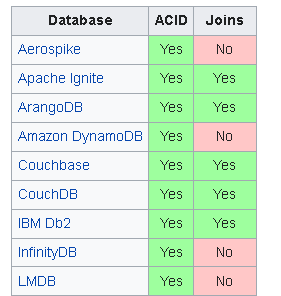
Một vài loại của NoSQL:

Key-value store sử dụng một mảng kết hợp như là mô hình dữ liệu cơ bản của chúng. Trong mô hình này, những dữ liệu được biểu diễn như là một tập hợp của key – value.

Document store có ý tưởng trung tâm là “document”. No sẽ đóng gói và mã hóa dữ liệu trong các format tiêu chuẩn hoặc tiêu chuẩn mã hóa. Việc mã hóa sử dụng bao gồm các định dạng như XML, JSON, YAML và dạng nhị phân như BSON. Những Document sẽ được xử lý trong cơ sở dữ liệu thông qua unique key.

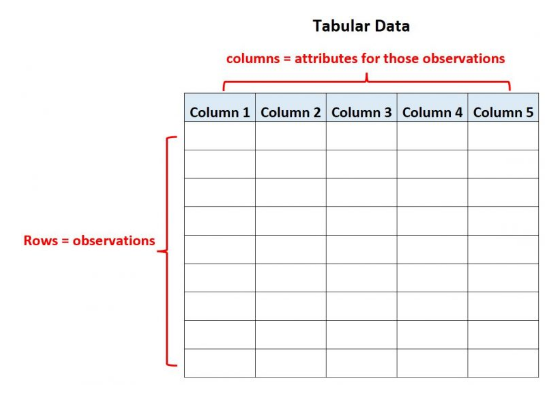
Graph là cơ sở dữ liệu được thiết kế cho dữ liệu mà quan hệ của nó được biểu diễn như một đồ thị chứa những thành phần được kết nối bởi con số hữu hạn của những quan hệ. Ví dụ của loại dữ liệu này là biểu thị tương quan xã hội, tuyến đường phương tiện giao thông công cộng, biểu đồ đường, cấu trúc mạng,….

ACID (atomicity – nguyên tử số, consistency – nhất quán, isolation – độc lập, durability – bền bỉ) là một tập hợp các thuộc tính của các chuyển giao cơ sở dữ liệu nhằm đảm bảo tính hợp lệ của dữ liệu khi có lỗi.



* 1. Dữ liệu dạng bảng

Trong thống kê, dữ liệu dạng bảng đề cập đến việc dữ liệu được sắp xếp trong một bảng với những hàng và cột.



Hình 4 – Dữ liệu dạng bảng

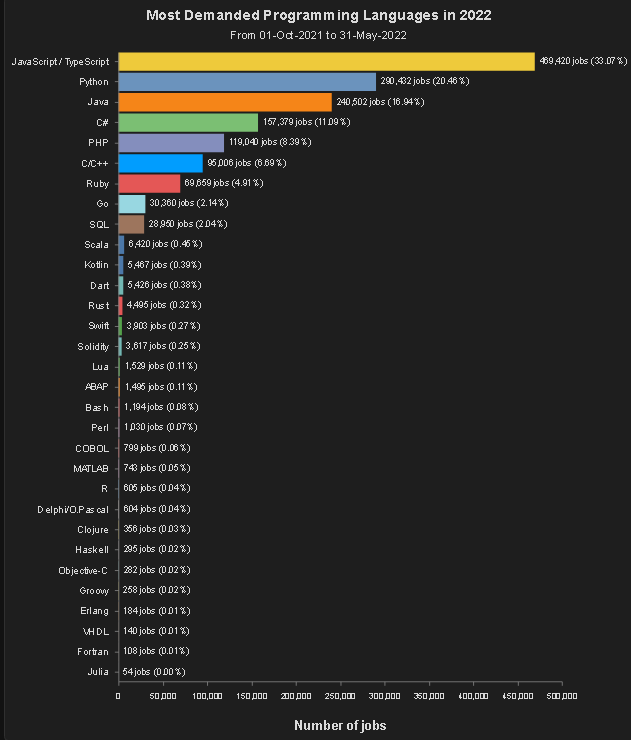
Trong thực tế, hầu hết dữ liệu được lưu trong spreadsheet của Excel bởi vì nó được xem như là dữ liệu dạng bảng. Excel là một ví dụ về dạng bảng đặc trưng, vì nó chứa nhiều hàng và cột.

* 1. Giới thiệu ngôn ngữ python
     1. Khái niệm

Python là một ngôn ngữ lập trình hướng đối tượng phổ biến. Nó được tạo ra bởi Guido van Rossum và ra mắt năm 1991. Nó được sử dụng cho:

* Lập trình web (phía server)
* Lập trình phần mềm
* Toán học
* Xây dựng hệ thống

Python là ngôn ngữ rất mạnh, high-level, cách thức trình bày của ngôn ngữ python rất rõ ràng và minh bạch, thuận tiện cho việc lập trình những chương trình phức tạp.



Hình 5 – Ngôn ngữ lập trình được ưa chuộng trong năm 2022

* + 1. Python có thể làm gì?

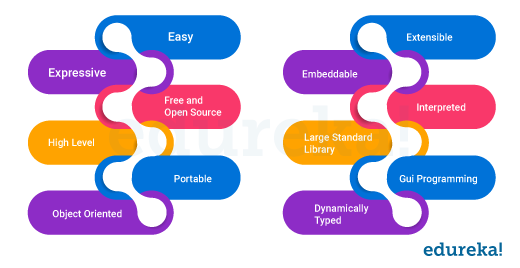
Python có thể được sử dụng trên một server để mà tạo những ứng dụng web. Bên cạnh đó, Python có thể dùng chung với phần mềm để mà tạo workflow. Python cũng có thể kết nối đến hệ thống cơ sở dữ liệu, có thể đọc và chỉnh sửa file,….



Hình 6 – Các ứng dụng của Python

* + 1. Tại sao lại chọn Python?

Python có thể hoạt động trên nhiều nền tảng (Windows, Mac, Linux, Raspberry Pi,..). Python có syntax đơn giản gần giống như là ngôn ngữ tiếng Anh, thư viện phong phú và cộng đồng sử dụng lớn.



Hình 7 – Các đặc điểm của Python

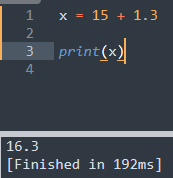
* 1. Cơ bản về lập trình Python
     1. Biểu thức (Expressions) trong Python

Biểu thức chính là một thực thể toán học. Gồm 2 phần chính:

* Toán hạng: có thể là một hằng số, biến số (X , Y).
* Toán tử: xác định cách thức làm việc giữa các toán hạng (+,-,\*,/).

Trong ngôn ngữ lập trình biểu thức được đánh giá theo mức độ ưu tiên của toán tử để mà nếu có nhiều hơn một toán tử trong biểu thức thì độ ưu tiên của chúng sẽ quyết định là hoạt động nào diễn ra trước. Chúng ta có rất nhiều loại biểu thức khác nhau trong Python.

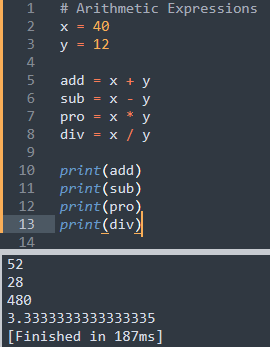
Biểu thức hằng số: Những biểu thức này chỉ có giá trị là hằng số.



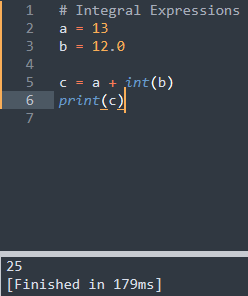
Biểu thức số học: Một biểu thức số học là sự kết hợp của những giá trị số, toán tử và một vài cặp ngoặc đơn. Kết quả của loại biểu thức này thì thường là giá trị số. Những toán tử được dùng trong biểu thức này thường là toán tử số học như phép cộng, phép trừ,….Dưới đây là một vài toán tử số học trong Python:

|  |  |  |
| --- | --- | --- |
| Operators | Syntax | Functioning |
| + | x + y | Addition |
| – | x – y | Subtraction |
| \* | x \* y | Multiplication |
| / | x / y | Division |
| // | x // y | Quotient |
| % | x % y | Remainder |
| \*\* | x \*\* y | Exponentiation |
|  |  |  |

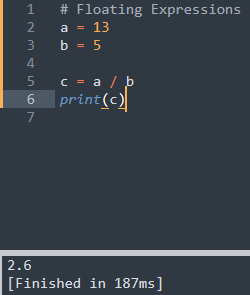
Bảng 1 – Các toán tử trong Python



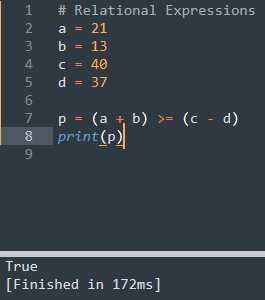
Biểu thức số nguyên: Những loại biểu thức này chỉ cho ra kết quả số nguyên sau khi mà tính toán và chuyển đổi loại.



Biểu thức số học: Những biểu thức này sẽ tạo ra kết quả số thực với dấu chấm sau khi mà tính toán và chuyển đổi loại.



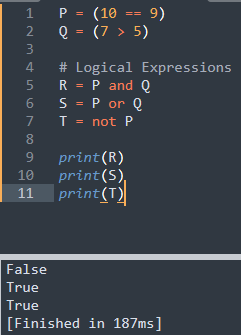
Biểu thức quan hệ: Trong những loại này của biểu thức, những biểu thức số học sẽ nằm cả bên của toán tử quan hệ (>, <, >=, <=). Những toán tử số học đó là được thực hiện phép tính trước sau đó sẽ đem đi so sánh theo toán tử quan hệ và cho ra kết quả boolean. Những biểu này cũng được gọi là biểu thức Boolean.



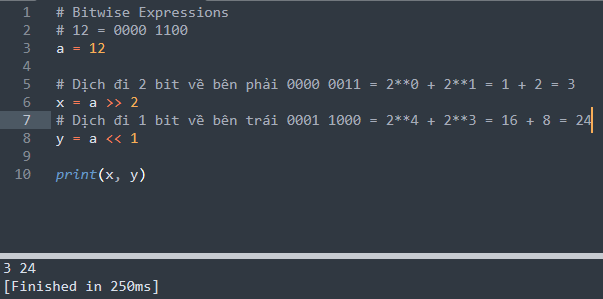
Biểu thức logic: Những biểu thức này sẽ trả về kết quả True hoặc False. Về cơ bản nó chỉ định một hoặc nhiều điều kiện. Ví dụ, (10==9) là một điều kiện nếu 10 bằng với 9. Như chúng ta đã biết thì điều đó là không đúng, vậy nên kết quả sẽ trả về False. Việc nghiên cứu biểu thức logic, chúng ta cũng sẽ chạm mặt với các toán tử logic. Dưới đây là một vài toán tử logic trong Python:

|  |  |  |
| --- | --- | --- |
| Operator | Syntax | Functioning |
| and | P and Q | It returns true if both P and Q are true otherwise returns false |
| or | P or Q | It returns true if at least one of P and Q is true |
| not | not P | It returns true if condition P is false |

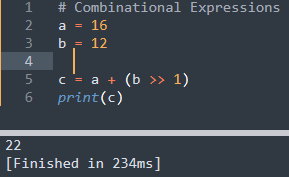
Bảng 2– Toán tử logic



Biểu thức Bitwise: Những biểu này sẽ tính toán dựa mức của bit (level)



Biểu thức kết hợp: Việc sử dụng những biểu thức khác nhau trong một biểu thức được xem như là biểu thức kết hợp.



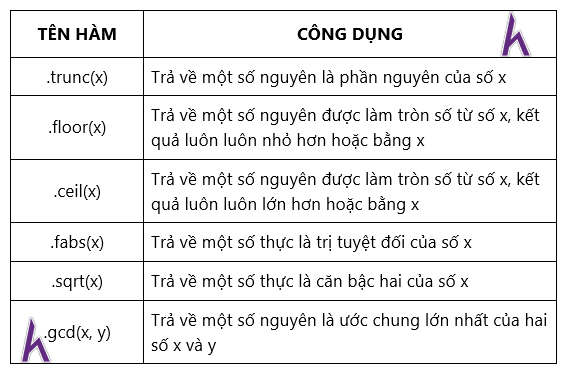
| Precedence | Name | Operator |
| --- | --- | --- |
| 1 | Parenthesis | ( ) [ ] { } |
| 2 | Exponentiation | \*\* |
| 3 | Unary plus or minus, complement | -a , +a , ~a |
| 4 | Multiply, Divide, Modulo | /  \*  //  % |
| 5 | Addition & Subtraction | +  – |
| 6 | Shift Operators | >>  << |
| 7 | Bitwise AND | & |
| 8 | Bitwise XOR | ^ |
| 9 | Bitwise OR | | |
| 10 | Comparison Operators | >=  <=  >  < |
| 11 | Equality Operators | ==  != |
| 12 | Assignment Operators | =  +=  -=  /=  \*= |
| 13 | Identity and membership operators | is, is not, in, not in |
| 14 | Logical Operators | and, or, not |

Khi chúng ta kết hợp nhiều loại biểu thức khác nhau hoặc sử dụng nhiều toán tử thì độ ưu tiên của toán tử sẽ phát huy tác dụng. Dưới đây là mức độ ưu tiên của toán tử:

Bảng 3 – Mức độ ưu tiên của các toán tử

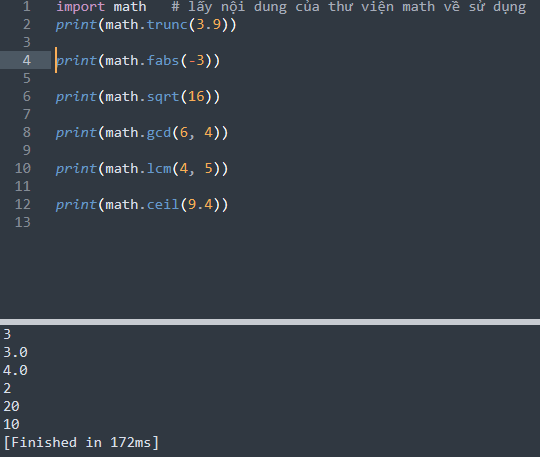
Thư viện math trong Python:

Trong Python thư viện math cung cấp các hàm dùng cho toán học. Bảng sau đây liệt kê một số hàm toán học hay được sử dụng.



Bảng 4 - Các hàm trong thư viện math

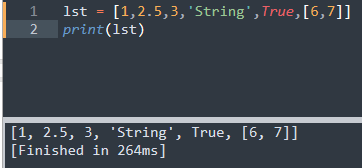
Ví dụ:



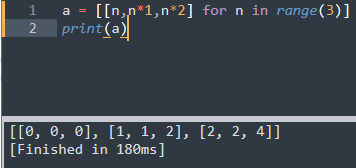
* + 1. Cấu trúc dữ liệu (Data Structure)
       1. List

List là một container được dùng phổ biến trong chương trình Python. List được giới hạn bởi dấu ngoặc vuông ‘[ ]’, các phần tử trong List được ngăn cách nhau bởi dấu phẩy. List có thể chưa mọi loại dữ liệu bao gồm cả chính nó.

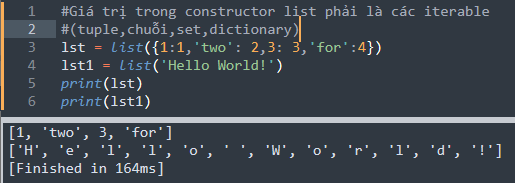
Khởi tạo List:



Dùng List comprehension để khởi tạo:

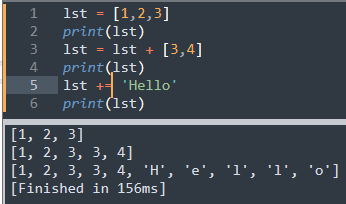


Dùng constructor list:

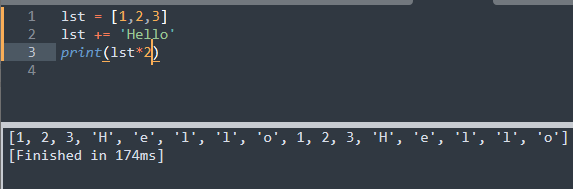


Các toán tử trong List:

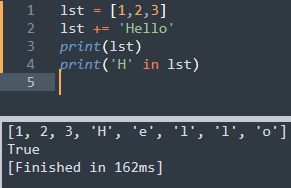
Toán tử +:



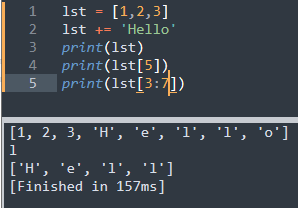
Toán tử \*:

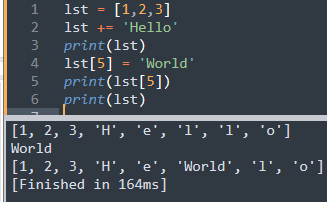


Toán tử in:



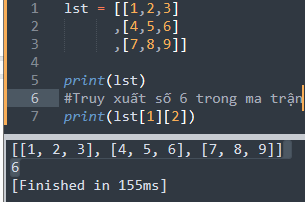
Indexing và cắt List:



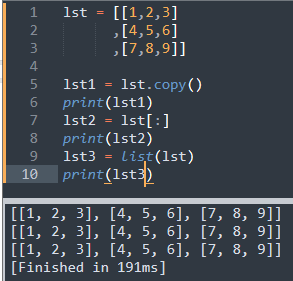


Ma trận:

Ma trận được tạo bởi việc một list chứa các list trong nó.



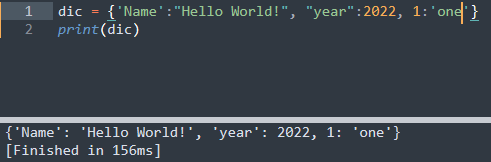
Khi mà cần sao chép một List ta không được gán một List đã có cho một List trực tiếp mà phải dùng các phương pháp khác nhau để sao chép. Việc này là để tránh khi ta thay đổi nội dung của List cũ thù List bị gán cũng sẽ thay đổi theo, tức là cả hai đối tượng đang trỏ về cùng một vùng nhớ.



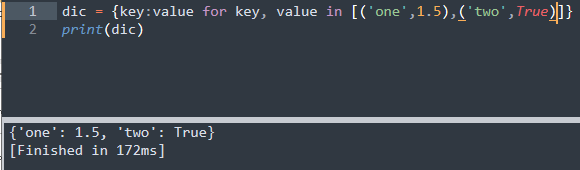
* + - 1. Dictionary

Dictionary hay gọi tắt là Dict cũng là một container như List. Khác với các container thông thường thì Dict sẽ dùng key để phân biệt các phần tử thay vì dùng index. Một Dict sẽ bị giới hạn bởi cặp dấu ngoặc nhọn ‘{ }’, được ngăn cách với nhau bởi dấu phẩy. Tuy nhiên các phần tử của Dict sẽ là một cặp key:value.

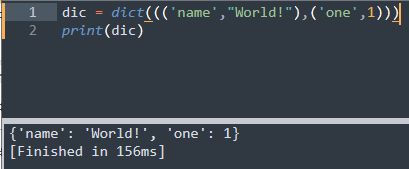
Khởi tạo Dict:



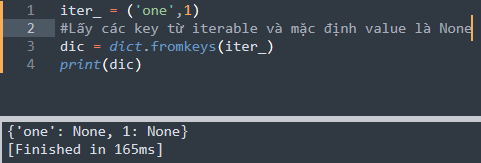
Sử dụng Dict comprehension để khởi tạo:



Khởi tạo bằng constructor:

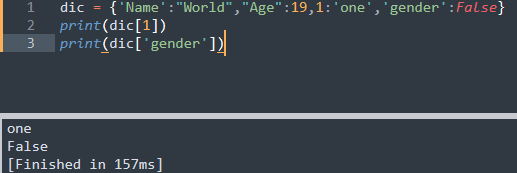


Khởi tạo bằng phương thức fromkeys:

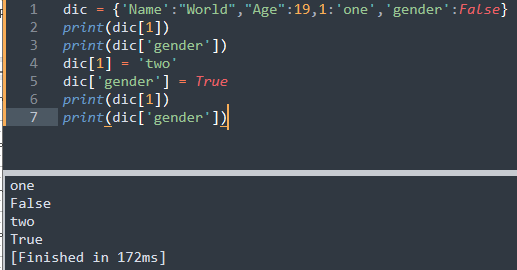


Truy xuất phần tử trong Dict:

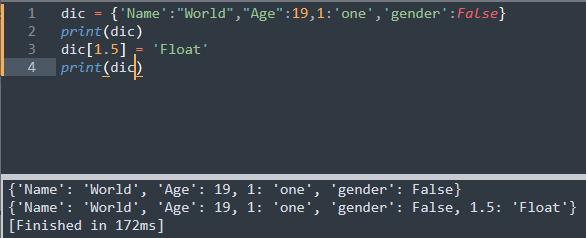
Khi truy xuất giá trị của key ta chỉ cần tham chiếu đến key mà ta muốn lấy giá trị.



Ta có thể thay đổi nội dung trực tiếp cho Dict giống List

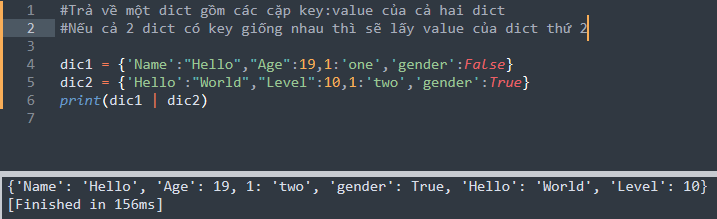


Ta cũng có thể thêm một phần tử vào Dict.



Toán tử trong Dict:

Toán tử |:



* + 1. Hàm (Function)

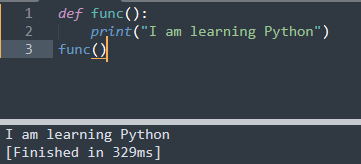
Hàm được hiểu là một phần đoạn code sẽ chạy khi mà được gọi, thường là đoạn code đó được dùng nhiểu lần trong chương trình. Ta cũng có thể gọi nó là method hoặc procedure. Python hỗ trợ rất nhiều hàm được xây dựng sẵn như là print(), input(), compile(),….Python cũng cho phép việc tạo ra một hàm do người dùng tự định nghĩa.

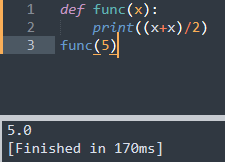
Để khai báo một hàm sử dụng từ khóa ‘def’ , cú pháp:

def <tên hàm> (parameter 1, ….parameter n):

function – block:

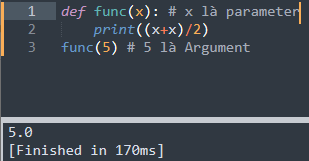
Ví dụ:



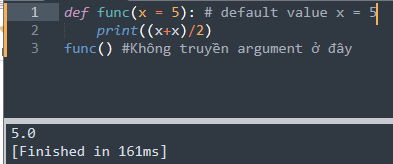


Parameter và Argument:

Parameter là các biến hay các tham số ta định nghĩa trong lúc khai báo hàm giới hạn bởi cặp dấu ngoặc ( ), còn Argument là giá trị mà ta gán cho biến đó khi gọi tên hàm.

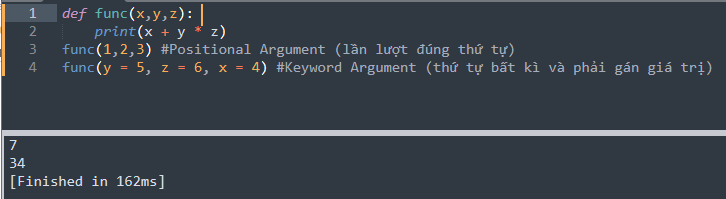


Ta có thể đặt giá trị mặc định cho parameter, trong trường hợp đó khi ta không đưa argument lúc gọi hàm thì hàm sẽ dùng giá trị mặc định mà ta đã gán.



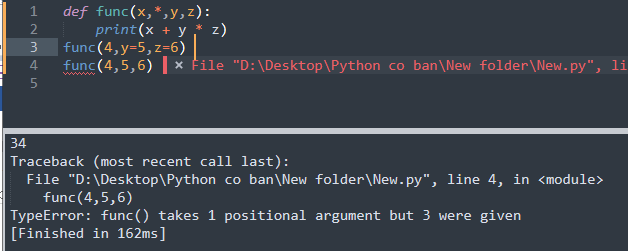
Phân biệt Positional Argument và Keyword Argument:

Việc ta đưa giá trị vào cho hàm lần lượt theo thứ tự của các parameter được hiểu như là positional argument nhưng bên cạnh đó ta có thể gán giá trị cho một parameter lúc ta gọi hàm và chúng được hiểu là keyword argument.



Lưu ý: Ta có thể truyền đồng thời cả positional argument và keyword argument. Tuy nhiên thì các positional argument bắt buộc phải đứng trước cái keyword argument.

Python có hỗ trợ cho phép ta buộc phải sử dụng kiểu keyword argument để truyền argument vào cho hàm. Cú pháp: def <tên hàm> (\*, argument 1, …, argument n). Các argument sau dấu \* buộc phải truyền theo kiểu keyword argument.



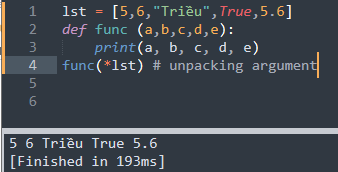
Ta cũng có thể làm điều tương tự cho positional argument với cú pháp như sau: def <tên hàm> (argument 1, …, argument n, /). Các argument đứng trước dấu / buộc phải là positional argument.



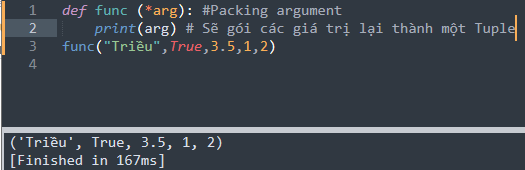
Packing Argument và Unpacking Argument:

Ta có thể hiểu packing là gói các giá trị vào trong một vùng nhớ duy nhất còn unpacking là ta lần lượt giải phóng từng giá trị ra các vùng nhớ khác nhau.

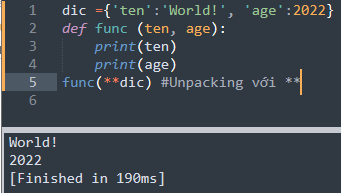
Sử dụng dấu \* để unpacking argument tức là khi ta có một container hay một iterable ta để dấu \* trước chúng khí gọi hàm để unpacking.



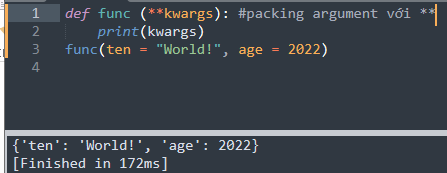
Ta cũng có thể dùng dấu \* cho việc packing argument khi ta muốn truyền vào nhiều giá trị nhưng chỉ khai báo một parameter.



Bên cạnh đó ta có thể dùng cả hai dấu \*\* để unpacking cũng như packing cho argument. Điều này thường dùng cho kiểu dữ liệu Dict bởi vì Dict đi với các cặp name:value vậy nên khi dùng một dấu \* ta chỉ có thể lấy được key.

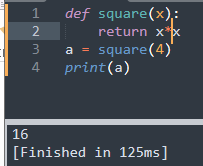


Lưu ý: khi ta unpacking cho dic với dạng này tức ta đang truyền vào argument dưới dạng keyowrd argument vậy nên tên parameter phải giống với key của Dict.

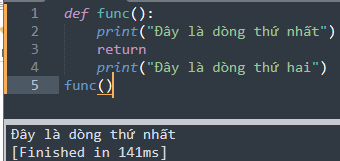


Kết quả của việc packing này sẽ cho ta kết quả của một Dict.

Trong trường ta muốn lưu kết quả xử lý của một hàm ra một biến thì Python hỗ trợ lệnh return để trả về giá trị khi các dòng lệnh trong hàm được chạy xong. Cú pháp: return object.



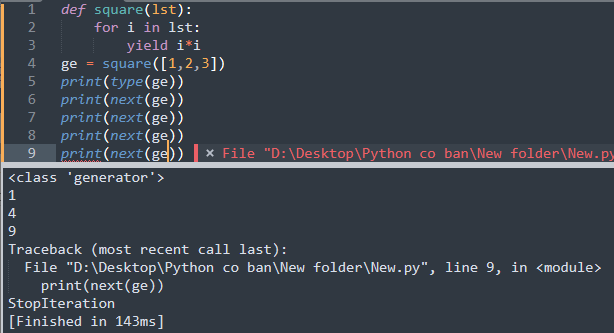
Chúng ta cũng có thể sử dụng lệnh return để mà ngắt dòng lệnh trong hàm.



Ta cũng có thể trả về nhiều hơn một giá trị, bản chất của nó giống như ta đang unpack từ một container.



Bên cạnh return thì Python cũng hỗ trợ kiểu trả về là một generator hay một iterator. Interator thường được so sánh với iterable, trong khi iterable cho phép bạn tái sử dụng truy xuất trực tiếp từng phần tử giống như List, Tuple, String,…nhưng iterator không cho phép điều đó mà phải truy xuất lần lượt từng phần tử với một hàm hỗ trợ là hàm next(). Khi ta truy xuất hết iterator thì ta sẽ không gọi lại được nữa. Cú pháp yield object.



1. MÁY HỌC – MACHINE LEARNING
   1. Khái niệm
      1. Machine learning

Machine learning là một hệ thống của những thuật toán máy tính có thể học từ những mẫu dữ liệu qua sự cải thiện và rút kinh nghiệm của chính hệ thống mà không có sự can thiệp của lập trình viên. Machine learning là một phần của trí tuệ nhân tạo, nó kết hợp dữ liệu và các công cụ thống kê để dự đoán để dự đoán kết quả hay đầu ra giúp thông tin trở nên linh hoạt. Machine learning có quan hệ mật thiết tới mô hình khai thác dữ liệu và dự đoán Bayesian. Máy tính sẽ nhận dữ liệu như đầu vào và sử dụng các thuật toán để mô phỏng câu trả lời.

Machine learning thực chất là việc lập trình những máy tính để tối ưu hóa tiêu chí hiệu suất bằng cách sử dụng những dữ liệu mẫu hoặc dữ liệu quá khứ. Ta có một mô hình được định nghĩa bằng một vài parameter và việc học (learning) là sự thực thi của một chương trình máy tính để tối ưu những parameter đó qua việc sử dụng liệu. Mô hình sẽ đưa ra dự đoán cho tương lai. Lĩnh vực nghiên cứu machine learning thì bận tâm với những câu hỏi như làm thế nào để xây dựng một chương trình máy tính có khả năng tự cải thiện với những kinh nghiệm mà máy có được.

Cấu trúc: Một chương trình máy tính được nói là học từ kinh nghiệm E với loại nhiệm vụ A và độ chính xác P nếu độ chính xác của nó ở nhiệm vụ T được đo bằng P và cải thiện theo kinh nghiệm E.

Ví dụ: Vấn đề nhận dạng chữ viết tay.

Nhiệm vụ A: Phân biệt các từ viết tay có trong ảnh.

Độ chính xác P: Phần trăm của những từ được phân loại một cách chính xác.

Kinh nghiệm E: Một bộ dữ liệu của những từ viết tay với các thông số được chỉ định.

Một nhiệm vụ điển hình của machine learning là cung cấp đề xuất. Đối với những ai mà có một tài khoản Netflix thì đề xuất của những bộ phim hoặc seri phim là được dựa trên dữ liệu lịch sử của người dùng. Những công ty công nghệ đang sử dụng việc học không giám sát (unsupervised learning) để cải thiện trải nghiệm người dùng với việc cá nhân hóa đề xuất.

* + 1. Phân loại

Supervised learning – Học giám sát là một thuật toán sử dụng dữ liệu train và phản hồi từ con người để học mối liên hệ giữa đầu vào và đầu ra. Ta có thể sử dụng supervised learning khi dữ liệu đầu ra đã được biết trước. Thuật toán sẽ dự đoán dữ liệu mới dựa vào các kết quả đã có.

Ví dụ: Dữ liệu dưới đây là về những bệnh nhân đi vào một phòng khám. Dữ liệu chứa tuổi và giới tính của mỗi bệnh nhân với kết quả là khỏe mạnh hoặc bệnh.

|  |  |  |
| --- | --- | --- |
| Giới tính | Độ tuổi | Tình trạng sức khỏe |
| Nam | 48 | Bệnh |
| Nam | 67 | Bệnh |
| Nữ | 53 | Khỏe mạnh |
| Nam | 49 | Bệnh |
| Nữ | 32 | Khỏe mạnh |

Dựa vào bảng trên ta sẽ xây dựng một hàm số biểu diễn các cặp đầu vào (giới tính và độ tuổi) – đầu ra (tình trạng sức khỏe). Hàm số đó sẽ tự dự đoán đầu ra khi ta đưa một đầu vào bất kì trong tương lai.

Supervised learning được chia thành hai vấn đề là hồi qui (regression) và phân loại (classification).

Unsupervised learning – học không giám sát: Một thuật toán giúp bạn xử lí dữ liệu đầu vào mà không biết trước kết quả đầu ra. Ta có thể sử dụng unsupervised learning ta không biết cách để phân loại dữ liệu.

Ví dụ: Tương tự như ví dụ trên nhưng lúc này ta không có cột tình trạng sức khỏe nữa.

|  |  |
| --- | --- |
| Giới tính | Độ tuổi |
| Nam | 48 |
| Nam | 67 |
| Nữ | 53 |
| Nam | 49 |
| Nữ | 32 |

Unsupervised learning được chia thành hai vấn đề là phân cụm và không phân cụm.

* + 1. Ứng dụng

Tự động hóa: Máy học sẽ giúp tự động toàn bộ trong mọi lĩnh vực mà không cần sự can thiệp của con người. Ví dụ, robot sẽ vận hành các bước quan trọng trong kế hoạch sản xuất.

Tài chính: Máy học đang phát triển rất nhanh trong công nghiệp tài chính. Các ngân hành dùng máy học để tìm những mẫu trong dữ liệu và ngăn chặn các vi phạm.



Hình 8 – Ứng dụng của machine learning

Y học: Sử dụng nhận dạng hình ảnh, đưa ra chẩn đoán các bệnh.



Hình 9 - Ứng dụng của machine learning trong y học

Chặn quảng cáo: Nhờ máy học các thư spam sẽ được phân loại và chặn tránh phiền nhiễu cho user.



Hình 10 - Ứng dụng chặn spam

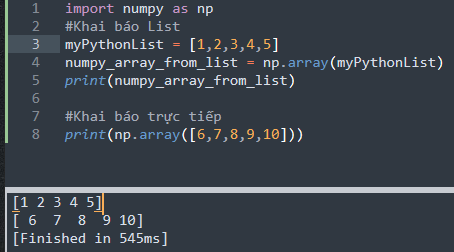
Tổ chức chính phủ: Chính phủ dùng máy học để quản lý các tiện ích và an toàn công cộng.

* 1. Thư viện Numpy

Numpy là thư viện nguồn mở sẵn có trong Python hỗ trợ trong việc tính toán, thống kê, kĩ thuật và lập trình khoa học dữ liệu. Nó rất có ích trong việc thực hiện các biểu thức toán học và thống kê, hoạt động rất hiệu quả cho những mảng đa chiều và việc nhân ma trận. Cho bất kì dự án khoa học nào thì Numpy là công cụ phải nên biết vì nó được xây dựng để làm việc với mảng N chiều, đại số tuyến tính, chuyển đổi Fourier,…, Numpy có khả năng xử lí lượng lớn dữ liệu có khả năng truy cập cao hơn các thư viện khác. Bên cạnh đó Numpy cũng rất tiện lợi cho việc nhân ma trận và sửa đổi ma trận.

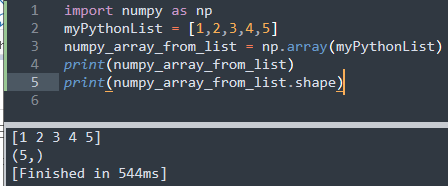
Một mảng Numpy là cấu trúc dữ liệu trung tâm của thư viện Numpy. Dưới đây chúng ta sẽ tìm hiểu cách tạo một mảnh bằng Numpy và các hàm hỗ trợ cho việc tính toán cái mà sẽ rất có ích khi ta học tới các thuật toán trong machine learning ở phần sau.

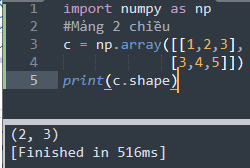
Tạo một mảng Numpy: Cách đơn giản nhất là dùng List trong Python hoặc dùng trực tiếp.

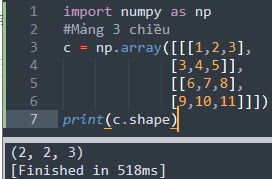


Bạn cũng có thể thực hiện các biểu thức toán học cơ bản như cộng, trừ, nhân, chia với ma trận với cú pháp: tên ma trận (+, -, \*, /).

Phần tử trong ma trận: Ta có thể kiểm tra số hàng và cột trong ma trận tùy thuộc vào số chiều của nó.







numpy.zeros(): Là hàm được dùng để tạo một ma trận với các phần tử 0, cú pháp:

numpy.zeros(shape, dtype=float, order=’c’).

shape: Số phần tử của mảng.

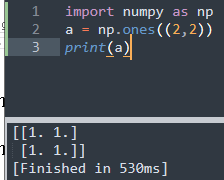
dtype: kiểu dữ liệu, mặc định là float.

order: mặc định là c.



numpy.ones(): Là hàm được dùng để tạo một ma trận với các phần tử 1, cú pháp:

numpy.ones(shape, dtype=float, order=’c’)



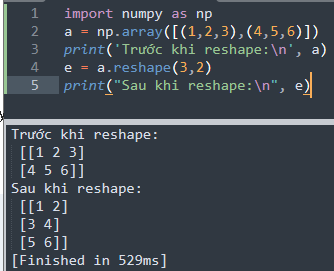
numpy.reshape(): Là hàm được dùng để định hình một mảng mà không làm thay đổi dữ liệu, cú pháp:

numpy.reshape(a, newShape, order=’C’)

a: mảng muốn định hình lại.

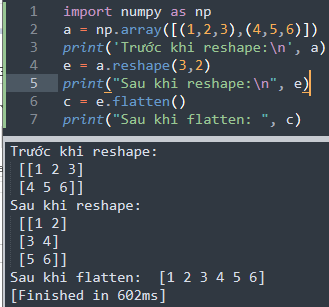
newShape: Hình dạng mong muốn.

order: Mặc định ‘C’.

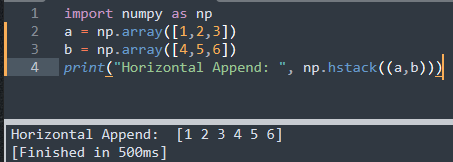


number.flatten(): Là hàm được dùng trả về một bản sao của mảng trong dạng một chiều, cú pháp:

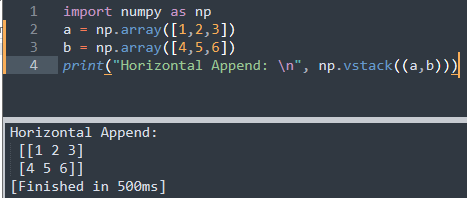
numpy.flatten(order=’C’).



numpy.hstack(): Là hàm được dùng để stack theo phương ngang tuần tự của mảng đầu vào để tạo thành một mảng đơn.



numpy.vstack(): Là hàm được dùng để stack theo phương dọc tuần tự của mảng đầu vào để tạo thành một mảng đơn.



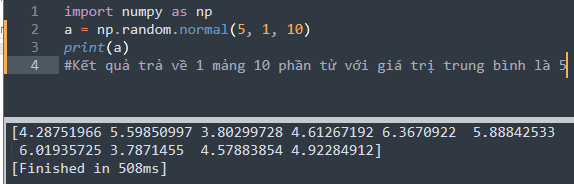
Tạo một mảng số ngẫu nhiên (phân phối chuẩn) ta sử dụng cú pháp:

Numpy.random.normal(loc, scale, size).

loc: giá trị trung bình, điểm trung tâm của sự phân phối.

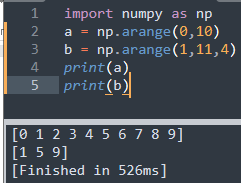
scale:độ lệch chuẩn.

size: số phần tử trả về.

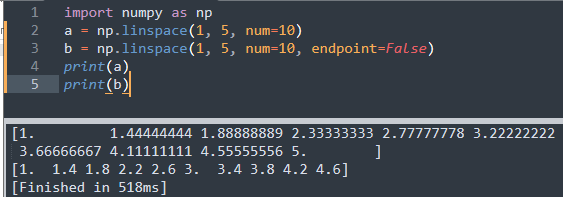


asarray(): Là hàm được dùng để chuyển đổi một đầu vào thành một mảng. Đầu vào có thể là List, Tuple, ndarray,….

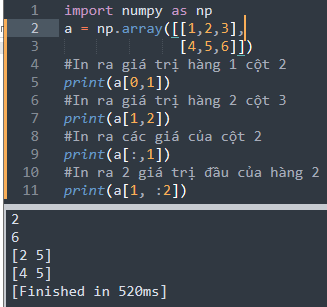
numpy.arange(): Là hàm trả về một ndarray object chứa các giá trị cách đều nhau trong một khoảng xác định, cú pháp cụ thể như sau: num.arange(start, stop, step, dtype). Trong đó start là khoảng bắt đầu, stop là khoảng dừng, step là khoảng cách giữa những giá trị mặc định là 1, dtype là loại dữ liệu đầu ra của mảng.



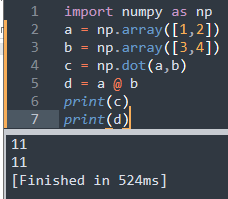
Numpy.linspace(start, stop, num, endpoint): là hàm tạo ra các phần tử cách đều nhau. Trong đó start và stop giống hàm arange(), còn num là số phần tử (mặc định là 50), endpoint mặc định là True để giá trị stop sẽ xuất hiện trong mảng.



Index và cắt ma trận:



numpy.dot(a, b): Trả về kết quả của phép nhân ma trận a và b, ta có thể sử dụng kí hiệu @.



* 1. Dữ liệu và xử lí dữ liệu

Dữ liệu có thể là dữ kiện bất kì như văn bản, hình ảnh, âm thanh chưa được phân tích và giải thích. Dữ liệu là phần quan trọng nhất trong tất cả từ phân tích dữ liệu (Data Analytics), Machine learning cho đến trí tuệ nhân tạo. Việc không có dữ liệu chúng ta sẽ không thể huấn luyện (train) bất kì mô hình nào (model), những nghiên cứu hiện đại hay các việc vận hành tự động sẽ trở nên vô ích. Có một sự thật không thể phủ nhận rằng các công ty hay các tập đoàn lớn đã chi trả rất nhiều tiền để thu thập các dữ liệu nhiều nhất có thể.

Thông tin là loại dữ liệu đã được phân tích, được vận dụng và trở nên có ý nghĩa cho người sử dụng.

Kiến thức là tổng hợp thông tin, kinh nghiệm, học tập và hiểu biết được suy luận. Là kết quả của sự nhận thức hoặc việc xây dựng những ý tưởng của một cá nhân hoặc tổ chức.



Hình 11 - Quá trình thay đổi của dữ liệu

Trong machine learning, dữ liệu thường được chia thành ba loại:

Dữ liệu số: Nếu một đối tượng mà được đo lường bằng những con số thì được gọi là dữ liệu số như số năm làm việc, mức lương,…

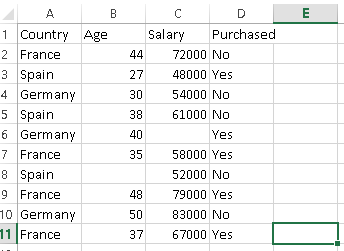
Dữ liệu phân loại: Là một thuộc tính được định tính trên một giá trị giới hạn và thường là cố định, nó còn được gọi là nominal. Ví dụ: True/False, Yes/No,….

Dữ liệu thứ tự: Được biểu thị trong một phạm vi có thứ tự. Ví dụ như kích cỡ quần áo sẽ có vừa, nhỏ, lớn hay mức độ hài lòng của khách hàng như không hài lòng, bình thường, hài lòng,…..

Xử lí dữ liệu là nhiệm vụ chuyển đổi dữ liệu từ dạng ban đầu thành dạng mong muốn và có nhiều khả năng sử dụng hơn, khiến nó có ý nghĩa và hữu ích. Bằng việc sử dụng những thuật toán của machine learning, mô hình toán học, kiến thức thống kê, toàn bộ quá trình chuyển đổi có thể tự động hóa. Đầu ra có thể là bất kì dạng dữ liệu nào mà ta mong muốn như hình ảnh, video, đồ thị, bảng,… Việc chuyển đổi này trông có vẻ đơn giản như khi đến những cái tổ chức lớn thì quy trình này cần thực hiện rất bài bản.

Ví dụ về dữ liệu và xử lí dữ liệu của machine learning bằng python:

Đầu tiên, chúng ta sẽ cần một bộ dữ liệu (dataset), dưới đây là dữ liệu dạng bảng thường được dùng trong machine learning và được lưu trữ trong file excel.



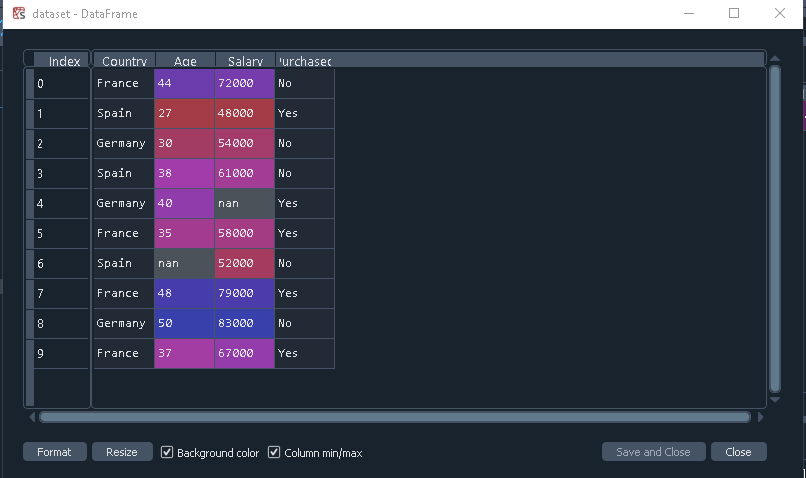
Như hình ta thấy ta có tổng cộng 4 cột và 10 hàng dữ liệu với các mục Country, Age, Salary, Purchased. Ở đây ta có 3 cột Country, Age, Salary là thông tin của một khách hàng và cột Purchased thể hiện khách hàng đó có mua sản phẩm của công ty hay không. Trong hầu hết mô hình machine learning là việc ta dự đoán một thông tin dựa trên thông tin đã có, vậy nên ta phải phân biệt hai dạng biến là independent variables và dependent variables, 3 cột đầu tiên chính là independent variables và cột Purchased chính là dependent variable. Việc mua sản phẩm hay không sẽ phụ thuộc vào các cột thông tin trước đó.

Để có thể xử lí dữ liệu ta cần lấy dữ liệu từ file excel trên. Chúng ta phải sử dụng một vài thư viện xử lý như numpy, ngoài ra python cũng có hỗ trợ một số thư viện phổ biến được dùng trong machine learning để việc vận dụng dữ liệu mượt mà hơn.

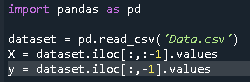
Để lấy dữ liệu ta import thư viện pandas.



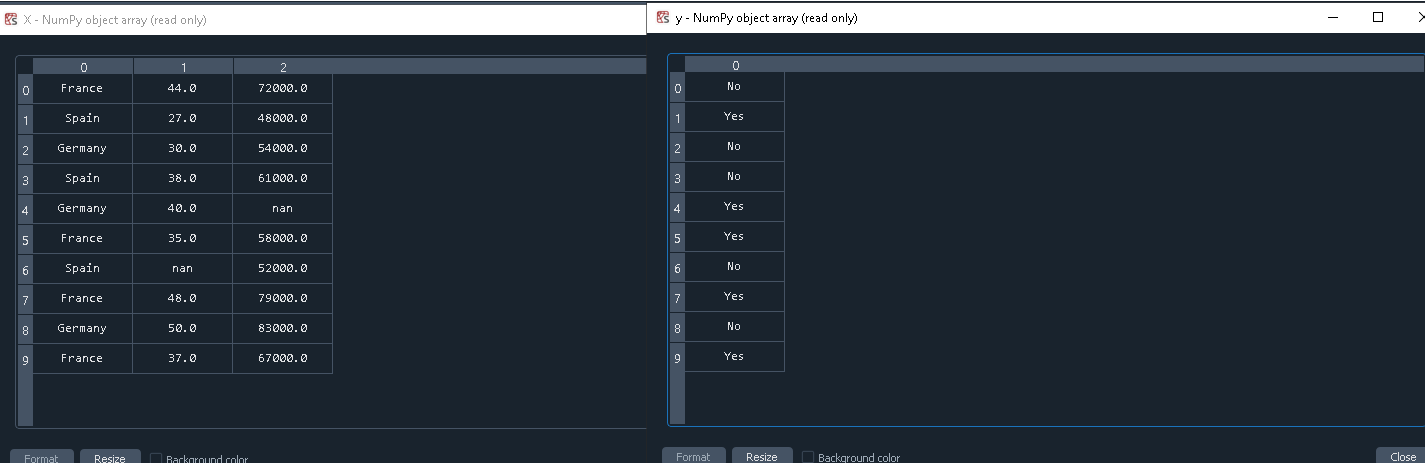
Dataset:



Tiếp đến như đã đề cập ở trên ta phải xử lí dữ liệu bằng cách lấy các thông tin trong các cột cụ thể là independent variables và dependent variable để sử dụng trong các mô hình của machine learning.

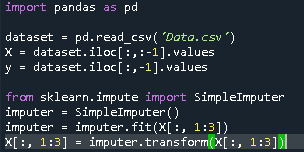


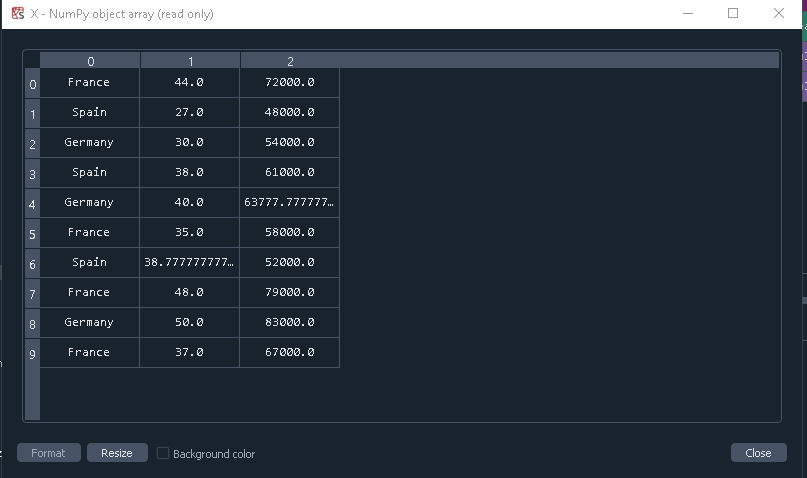
Ta tách thành hai ma trận riêng biệt:



Trong bảng dữ liệu này ta thấy có một số dữ liệu bị mất hoặc thiếu được thay thế bằng chữ ‘nan’. Đó là lí do tại sao ta cần phải xử lí dữ liệu, một phương pháp được đưa ra ở đây là những giá trị bị mất ở bản trên sẽ được tính bằng giá trị trung bình của cột tương ứng.

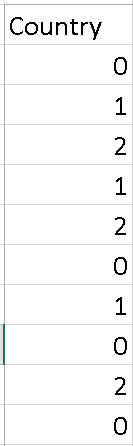
Python hỗ trợ ta một thư viện để làm điều đó:





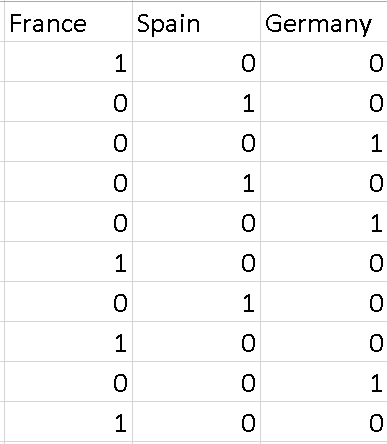
Trong bộ dữ liệu ta có hai cột là dữ liệu phân loại là cột Country và cột Purchased. Như ta đã biết, machine learning sẽ làm việc với các biểu thức toán học do đó việc giữ các giá trị phân loại là gây lỗi trong quá trình tính toán vì nó là ở định dạng chữ chứ không phải ở dạng dữ liệu số. Đó là lý do ta sẽ mã hóa hai cột này để đưa vào xử lí các thuật toán xa hơn. Ta sẽ phân tích hai kĩ thuật khác nhau được dùng trong Python để làm việc này là Label Encoding và One Hot Encoding.

Label Encoding: Đề cập đến việc chuyển những nhãn thành một dạng số để mà máy có thể đọc được. Ví dụ ta có cột Country trong dataset:

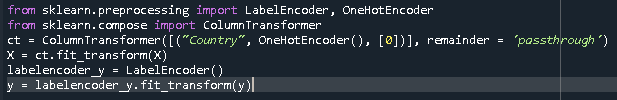
 

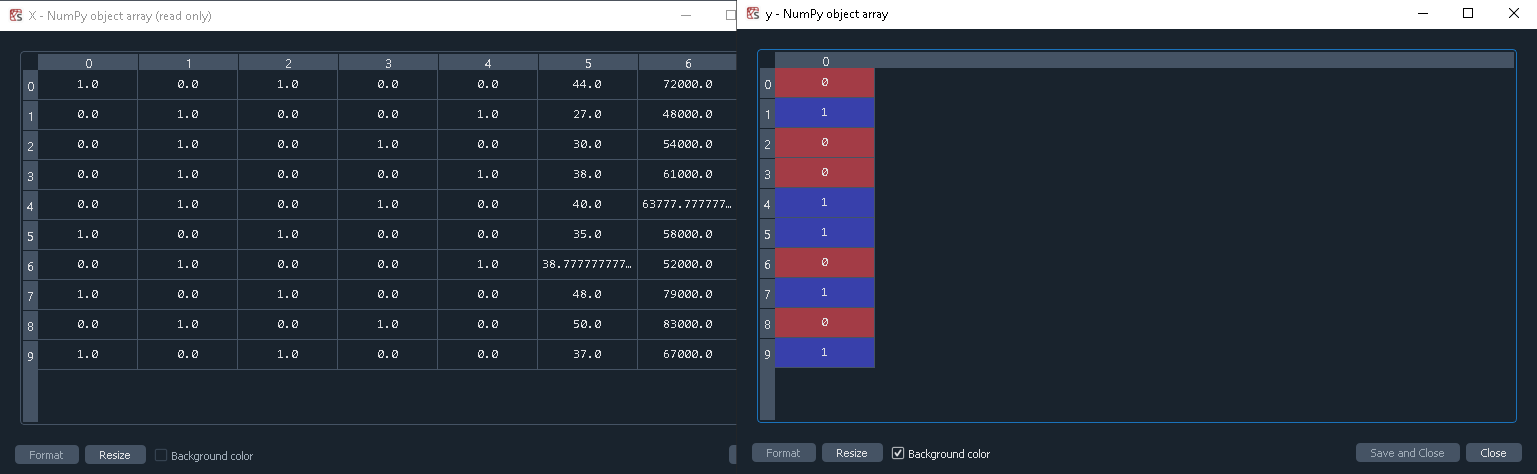
Có một giới hạn khi sử dụng kĩ thuật này là tạo ra các vấn đề liên quan đến độ ưu tiên, máy sẽ hiểu là nhãn nào được đánh số lớn hơn sẽ có độ ưu tiên cao hơn như trong hình thì Germany sẽ có ưu tiên cao hơn France và Spain nhưng thực chất không có mối quan hệ ưu tiên giữa các nước ở đây. Chính vì vậy nó có thể dẫn tới những kết quả sai lệch, do đó kĩ thuật One Hot Encoding được sử dụng để giải quyết vấn đề này.

One Hot Encoding: Trong kĩ thuật này, những dữ liệu phân loại sẽ được biểu diễn thành các cột riêng lẻ. Vậy nên, khi nào có France thì giá trị 1 sẽ được đánh cho cột France, giá trị 0 cho hai cột Spain và Germany tương tự như thế cho cả 2 cột nếu xuất hiện.

Dưới đây là dữ liệu sau khi ta mã hóa, ta sẽ dùng One Hot Encoding cho cột Country và Label Encoding cho cột Purchased bởi vì giá trị yes và no về mặc logic thì yes sẽ có ưu tiên cao hơn.



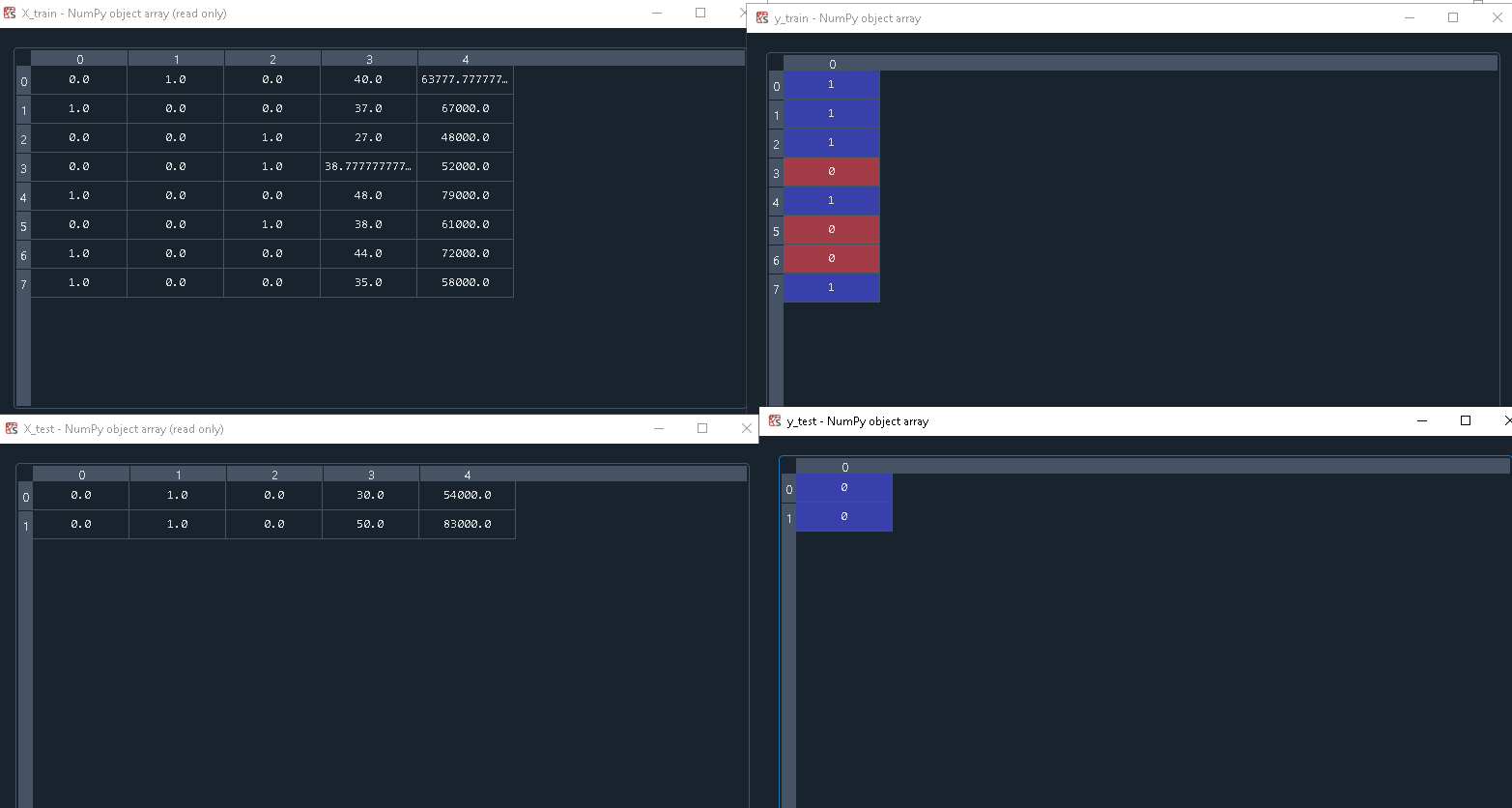


Mục tiêu của machine learning là sẽ học và làm một thứ gì đó trên bộ dữ liệu mà nó được cấp bằng cách hiểu sự tương quan giữa các dữ liệu, khi máy chỉ học nhiều về các mối tương quan với dữ liệu, ta sẽ không chắc được hiệu năng của việc học trên bộ dữ liệu đó. Do đó, ý tưởng tách bộ dữ liệu thành hai phần riêng biệt đã được thực hiện trong machine learning. Ta sẽ chia bộ dữ liệu thành bộ dùng để train và để test, khi ta test trên cùng một bộ train hiệu năng sẽ được đảm bảo, tạo sự chắc chắn khi ta cấp một bộ dữ liệu hoàn toàn mới trên mô hình ta đã xây dựng trước đó.

Để làm điều này trong python ta dùng câu lệnh sau:



Với thông số lần lượt là bộ dữ liệu muốn tách, tỉ lệ tách (thường là 25% hoặc 20%), thông số random\_state để mỗi lần chạy code bộ dữ liệu sẽ không bị tách lại ngẫu nhiên. Với dòng code trên ta sẽ được 8 hàng đầu sẽ được dùng cho bộ train và 2 hàng cuối sẽ dùng cho bộ test.



* 1. Học giám sát (Supervised Learning)
     1. Hồi quy tuyến tính (Linear Regression)

Linear Regression là một thuật toán machine learning thuộc nhóm supervised learning. Regression mô hình hóa một giá trị dự đoán mục tiêu dựa trên independent variables. Thường được dùng cho việc tìm ra mối quan hệ giữa giả thiết – kết quả. Linear Regression thực hiện nhiệm vụ dự đoán dependent variable (y) dựa trên independent variables (x). Vì kĩ thuật này sẽ tìm ra một mối quan hệ tuyến tính giữa x và y nên nó có tên là Linear Regression.

Công thức:

Đối với dữ liệu chỉ có một feature hay một independent varibale (univeriate linear regression) thì hàm dự đoán là:­

ypred = θ­­­0 + θ1x1.

Đối với dữ liệu có nhiều feature hay independent variables (multiveriate linear regression) thì hàm dự đoán là:

ypred = θ­­­0 + θ1x1 + θ2x2 +… + θnxn.

Trong đó:

x1, x2, …, xn : là các dữ liệu đầu vào tương ứng.

θ­­­0, θ1,θ2, θn: Là các parameter để điều chỉnh giúp hảm số đưa ra kết quả dự đoán gần với giá trị thực nhất.

ypred: là kết quả dự đoán.

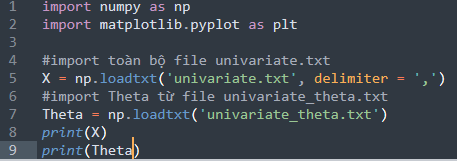
Ta có thể thấy rằng hàm dự đoán khá giống với việc ta nhân ma trận với vector, giả sử ta có: = và ma trận X = và hàm dự đoán sẽ là một ma trận chứa các giá trị dự đoán ypred = X\* . Để có thể nhân hai ma trận thì số cột của X phải bằng số hàng của , với θ­­­0 ta sẽ không có x0 do đó ta sẽ thêm một cột giá trị 1 vào ma trận X.

X = , kết quả ypred = .

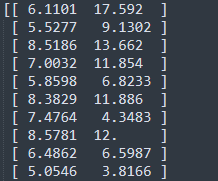
Từ hàm ypred = θ­­­0 + θ1x1 ta có thể mở rộng ra các hàm như mũ hai hoặc là hàm lấy căn. Như ta đã biết machine learning sẽ làm việc với một bộ dữ liệu thường là dạng bảng, vậy nên ta cần phải làm quen với các phép toán về ma trận cũng như là các phép biến đổi tương đương.

Dưới đây là một ví dụ dùng python đọc bộ dữ liệu và đưa kết quả dựa trên hàm dự đoán chúng ta tạo. Các parameter giả sử đã được chọn sẵn (không chính xác 100%).

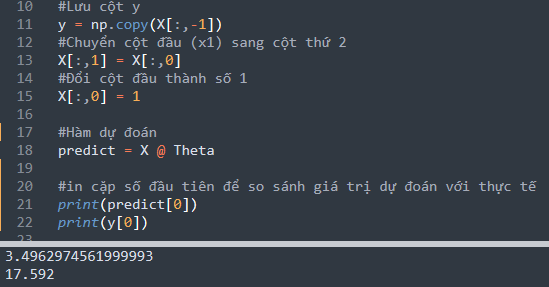
Đầu tiên ta import thư viện và dùng thư viện numpy để đọc dữ liệu từ file .txt:



Bộ dữ liệu gồm hai cột X, y (chỉ lấy minh họa vài hàng đầu) và parameter θ do ta tự đặt:

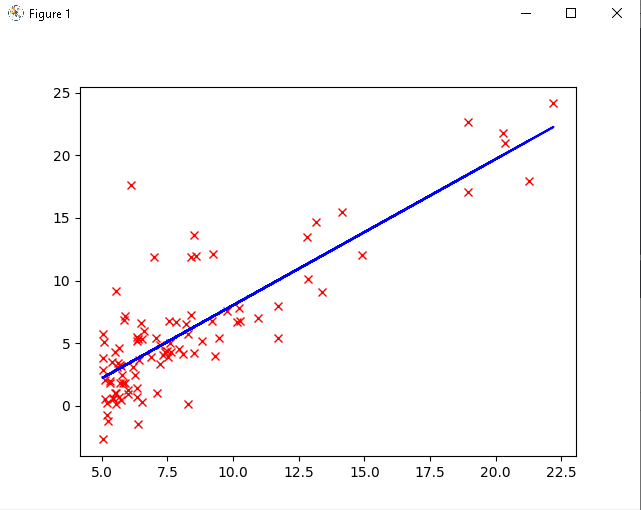
 

Lúc này ta tách cột y ra riêng và giữ lại để so sánh với hàm dự đoán. Còn cột X ta sẽ xử lý như việc giả sử ở trên, tức là thêm cột 1 vào trước cột X. Cuối cùng nhân ma trận với θ để ra hàm dự đoán.



Như ta thấy, giá trị dự đoán chưa chính xác với giá trị thực tế, ta có thể thay đôi θ để tăng độ chính xác. Cách tím θ tối ưu sẽ được giới thiệu ở phần sau.

Ta có thể vẽ đồ thị để mà quan sát mối quan hệ giữa các giá trị dễ dàng hơn.



Với các điểm x đỏ là các giá trị từ bộ dữ liệu và đường màu xanh chính là hàm dự đoán của chúng ta ở từng điểm được nối lại thành. Giống như phương trình đường thẳng.

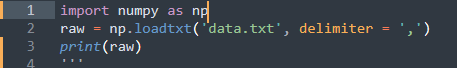
Để mà đo độ chính xác giữa giá trị dự đoán và giá trị thực tế ngoài việc là so sánh trực tiếp giá trị, ta xây dựng một hàm giúp ta dễ kiểm soát hàm dự đoán, khiến giá trị dự đoán gần với giá trị thực hơn mà không cần sự quan sát, giám sát của con người. Hàm đó được gọi là hàm tổn thất kí hiệu là J(θ).

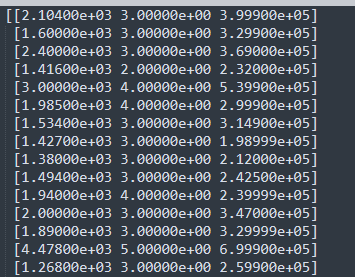
J(θ) = 2.

Hàm tổn thất sẽ cho ta biết độ lệch hay độ chính xác của hàm dự đoán hiện tại so với giá trị thực. Khi J(θ) = 0 tức là độ lệch giữa giá trị thực và giá trị dự đoán bằng 0, do đó hàm dự đoán lúc này sẽ là hàm chuẩn nhất.

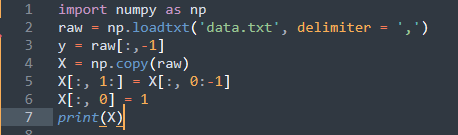
Ví dụ: Lập trình hàm tổn thất trong Python.

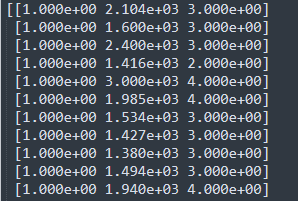
Đầu tiên ta cũng import thư viện và load bộ dữ liệu từ file .txt, trong file có ba cột tương đương hai cột đầu là independent variables và cột cuối là dependent varibale.



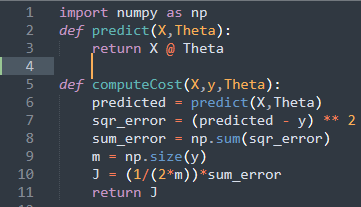


Tiếp đến ta thực hiện các bước như đã làm với hàm dự đoán ở ví dụ trên, tách cột cuối ra lưu lại, thêm một cột giá trị 1 vào trước hai cột feature.

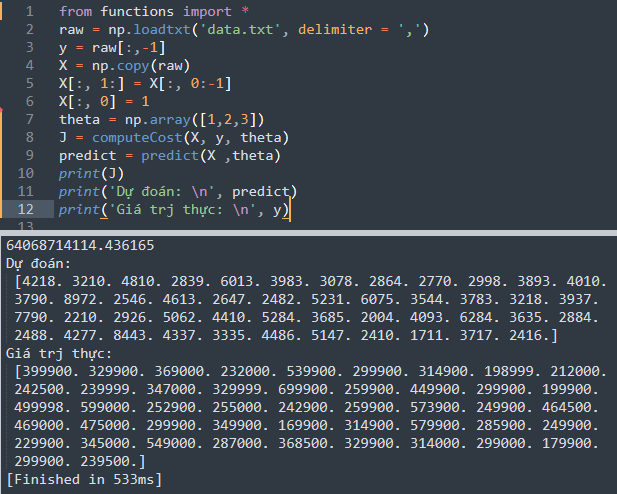




Tiếp đến ta sẽ xây dựng hàm J(θ), tên hàm là compute cost. Ta sẽ dùng một file khác để chứa hai hàm là hàm dự đoán (công thức từ ví dụ trước) và hàm J(θ).



Cuối cùng ta ta sẽ dùng test hàm J(θ) chúng ta vừa tạo từ bộ dữ liệu đã load vào. Parameter θ ta giả sử đã có sẵn (do ta tự chọn).



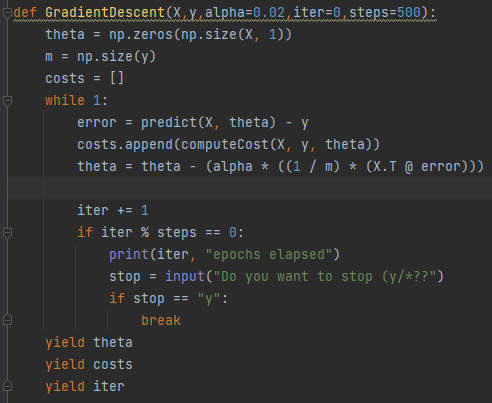
Như ta thấy kết quả hàm J(θ) rất lớn do đó hàm dự đoán sẽ cho kết quả sai lệch với giá trị thực từ bộ dữ liệu. Mục tiêu của chúng ta sẽ tối thiểu hóa hàm tổn thất để kết quả được sát với giá trị thực nhất.

Để giải quyết vấn đề trên thuật toán gradient descent được sử dụng để tìm ra θ hay chính là parameter ta đã đề cập ở trên. Nhiệm vụ chính của gradient descent sẽ giảm hàm J(θ) xuống nhỏ nhất. Theta tối ưu nhất mà tại đó biểu đồ của hàm J(θ) là thấp nhất. Vậy tức là ta sẽ cố gắng tìm ra bộ parameter θ tối ưu bằng cách thử nhiều bộ parameter sao cho J(θ) là nhỏ nhất, trong lập trình ta sẽ quyết chúng bằng vòng lặp.

Công thức gradient descent: θj = θj - α =θj - xj. Trong đó α là tốc độ học (learning rate) α càng lớn thì hàm J(θ) giảm càng nhanh. Đó là công thức để tìm từng θ (θ1, θ2, …, θn). Ta biết parameter θ sẽ dưới dạng một vector cho nên ta sẽ có công thức để tính dưới dạng vector là θ = θ - (XT \* (θ\*X – y)), lúc này y là một ma trận.

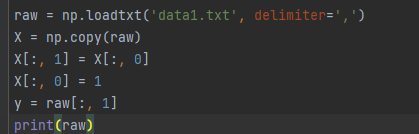
Bây giờ ta sẽ bắt tay vào xây dựng hàm gradient descent cho máy tính.

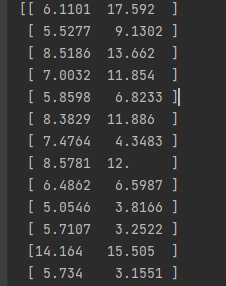
Dữ liệu chúng ta sẽ dùng chỉ có một feature vậy nên chỉ có θ0 và θ1 là hai parameter ta sẽ tìm. Giả sử ban đầu ta cho hai parameter này bằng 0 và giá trị sẽ được cập nhập sau mỗi vòng lặp.



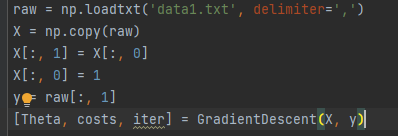
Theo đoạn code, ta sẽ có một list của hàm J(θ), mục đích là sau khi tìm được giá trị θ tối ưu ta sẽ dùng để vẽ đồ thị thuận tiện cho việc quan sát. Ta bắt đầu vòng lặp vô tận while, tính toán các thông số trong công thức của θ và sau mỗi 500 lần lặp (tức là iter ban đầu bằng 0 mỗi vòng lặp iter sẽ tăng lên 1 cho đến khi đủ steps=500). Chương trình sẽ hỏi ta có muốn dừng hay không, ấn ‘y’ để thoát và trả về giá trị, còn không sẽ tiếp tục lặp. Bằng cách in ra giá trị của J(θ) sau mỗi vòng lặp, ta có thể quan sát được khi nào J(θ) không thể nhỏ hơn được nữa, đó là lúc ta sẽ ấn ‘y’.

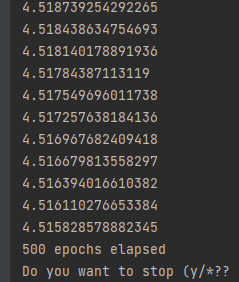
Ta sẽ test bộ dữ liệu dưới đây, quá trình load dữ liệu tương tự như các ví dụ trên.





Bây giờ, ta dùng hàm gradient descent ta đã dựng trước để lấy giá trị θ tối ưu.





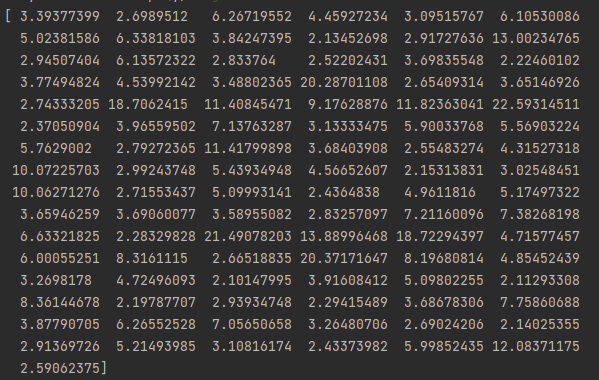
Giá trị của 500 J(θ) đầu tiên, ta thấy các J(θ) vẫn còn chênh lệch tức là vẫn còn giảm được.



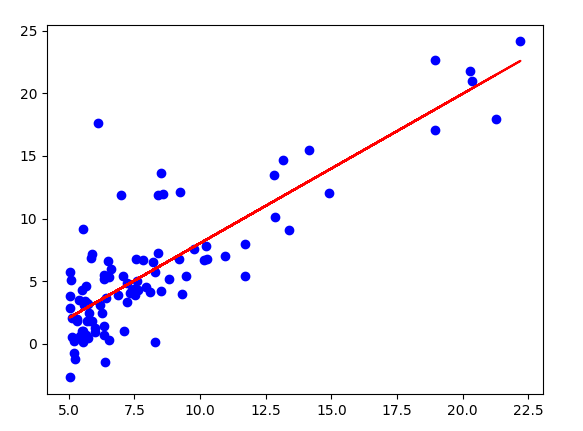
Sau 5500 vòng lặp ta thấy rằng các J(θ) đã dường như bằng nhau, chứng tỏ J(θ) không thể giảm thêm được nữa. Lúc này ta ấn ‘y’ để thoát và nhận giá trị θ tối ưu.



Khi đã có θ thì việc còn lại của chúng ta là dùng nó để tìm hàm dự đoán và đưa ra kết quả.



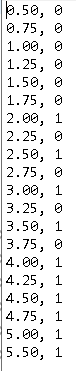
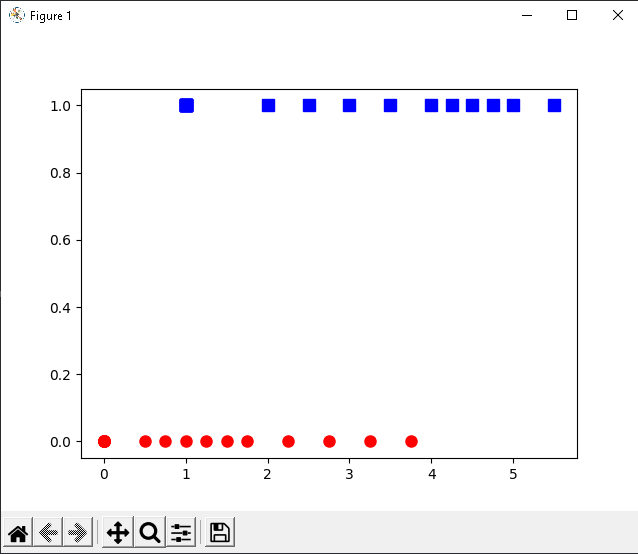
Đây chính là giá trị của hàm dự đoán và đồ thị tương ứng.



Lưu ý: Việc α là do ta tự chọn cho nên có thể sẽ không thật sự tối ưu, vì nếu dùng α quá lớn hàm J(θ) sẽ vượt qua khỏi điểm tối ưu còn nếu quá nhỏ thì sẽ xử lí rất chậm.

* + 1. Logistic Regression

Hãy cân nhắc một ví dụ về một bộ dữ liệu dưới đây, số giờ học tương ứng và kết quả của kì thi. Kết quả chỉ có hai giá trị là đậu (1) hoặc rớt (0).

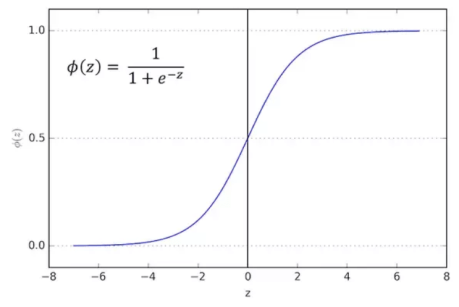
 

Rõ ràng ta thấy mô hình linear regression không còn phù hợp trong trường hợp này nữa, những giá trị kết quả là những giá trị rời rạc và được phân loại. Từ đó, ta sẽ cần một mô hình linh hoạt hơn đó là logistic regression. Ta cần một hàm số phải có tính chất sau:

* Là hàm số liên tục, nhận giá trị thực bị chặn trong khoảng (0, 1).
* Nếu điểm ngưỡng là điểm có giá trị tung độ 0.5 làm điểm phân chia thì các điểm càng xa điểm này về phía phải sẽ có giá trị bằng 1 và các điểm càng về phía trái có giá trị bằng 0.
* Có đạo hàm ở mọi điểm.

Dựa vào các tính chất trên ta có một hàm số thỏa mãn đó là hàm sigmoid.

g(z) =



Vậy là ta đã có hàm số phù hợp, bây giờ chúng ta sẽ bắt đầu xây dựng các hàm liên quan như đã làm trong bài toán linear regression, từ bộ dữ liệu ta có:

y =

Giả sử rằng bộ dữ liệu đầu vào có p cột feature và n hàng, ma trận X được biểu diễn như sau:

X =

Trong linear regression thì hàm dự đoán sẽ có dạng yi = θ0 + θ1xi1 + θ2xi2 + … + θpxip ở mỗi hàng với i nhỏ hơn hoặc bằng n. Hay ta có thể viết gọn thành dạng tổng quát cho ma trận y = θ\*X. Tới đây ta sẽ áp dụng hàm sigmoid vào.

y = g(θ\*X) =

Vậy là ta đã có hàm dự đoán cho thuật toán logistic regression, tiếp đến ta sẽ áp dụng xác suất để cho việc dự đoán chính xác hơn, giả sử mỗi hàng i của bộ dữ liệu có xác suất đậu và rớt như sau:

P(yi=1|xi;θ) = g(θ\*X)

P(yi=0|xi;θ) = 1 - g(θ\*X)

Ta có thể viết dạng gọn hơn P(yi|xi;θ) = g(θ\*X)yi(1 - g(θ\*X))1- yi. Chúng ta muốn mô hình gần với dữ liệu đã cho nhất, tức xác xuất phải đạt giá trị cao nhất.

Xét một ma trận feature X = [x1, x2,…, xn] và ma trận kết quả y = [y1, y2,…, yn] chúng ta sẽ tìm θ để hàm P(y|X;θ) đạt giá trị lớn nhất. Vì hàm xác xuất lúc này là các điểm dữ liệu xảy ra đồng thời, để đơn giản hóa bài toán ta sẽ giả sử các điểm này xảy ra độc lập với nhau:

P(y|X;θ) = P(yi|xi;θ) = g(θ\*X)yi(1 - g(θ\*X)) 1-yi

Bài toán này được gọi là maximum likelihood.

Vì trong hàm vẫn còn số mũ nên ta sẽ lấy log để thuận tiện cho việc tính toán và xây dựng thuật toán.

l(θ) = log(P(y|X;θ)) = g(θ\*X)) + (1-yi).log(1- g(θ\*X))

Khi N trở nên quá lớn thì việc tích của N phần tử sẽ là một số rất nhỏ dẫn đến sai số, để tránh điều đó ta sẽ lấy ngược dấu lúc này ta sẽ có hàm tổn thất cần tìm.

J(θ) = - g(θ\*X)) + (1-yi).log(1- g(θ\*X)))

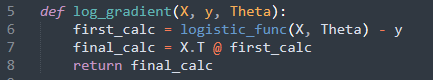
Vậy là ta đã có hàm tổn thất và giờ ta chỉ cần tìm bộ theta tối ưu để J(θ) về giá trị nhỏ nhất như trong linear regression, công thức tính theta θ = θ - α(XT\*(g(θ\*X) – y)).

Ta hãy bắt tay vào dự đoán bộ dữ liệu ví dụ ở lúc đầu với python.

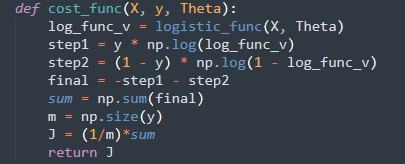
Trước tiên ta tạo hàm g(θ\*X) đạt tên là logistic\_func.

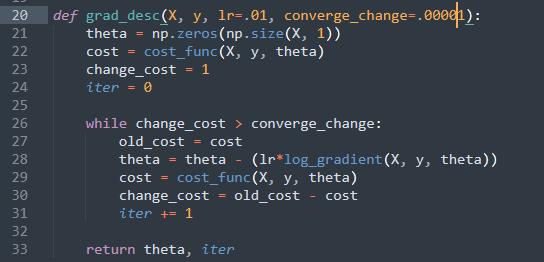


Kế tiếp ta tạo hàm log\_gradient để tính (g(θ\*X) – y) trong công thức theta.

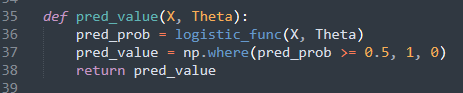


Viết hàm tính J(θ) và hàm tính theta như trong linear regression, ở đây ta xây dựng điều kiện dừng là khi hàm J(θ) của vòng lặp sau có sai số không lớn hơn 10-4 so với J(θ) của vòng lặp trước.



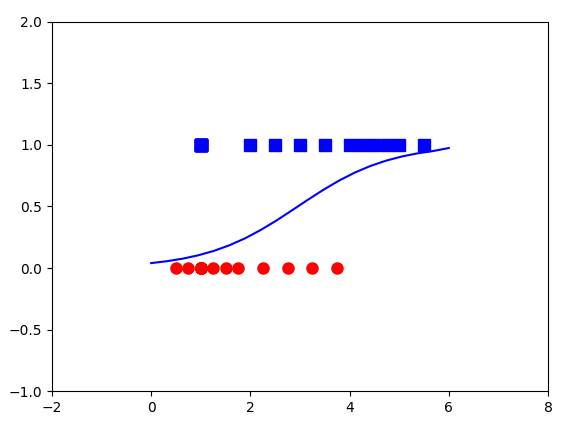


Và cuối cùng là hàm xác suất dự đoán (0 hoặc 1) dựa trên hàm sigmoid.



Bây giờ ta chỉ cần đưa bộ dữ liệu vào để xem kết quả, việc thiết lập các biến dependent và independent tương tự như cách làm trước, ở đây ta cũng sẽ minh họa trên trục tọa độ để theo dõi. Ta có thể thấy trong hàm xác suất có dòng lệnh np.where nghĩa là nếu giá trị từ hàm sigmoid mà lớn hơn hoặc bằng 0.5 thì sẽ tính là 1 (đậu) còn lại là rớt (0).





Từ kết quả được in ra ở trên ta thấy chỉ có 16 điểm dữ liệu là được dự đoán đúng, rõ ràng ta thấy nếu học sinh học 2 tiếng kết quả sẽ là đậu nhưng khi đưa vào hàm sigmoid kết quả nhỏ hơn 0.5 nên được phân loại là rớt, tương tự với các kết quả dự đoán sai còn lại. Cho nên việc lựa chọn mô hình phù hợp cho bộ dữ liệu là rất quan trọng.

* 1. Học không giám sát (Unsupervised learning)
     1. Phân cụm K-means

Chúng ta đã làm quen với hai thuật toán cơ bản trong học giám sát đó là Linear Regression và Logicstic Regression. Như ta đã biết trong hai thuật toán trên bộ dữ liệu luôn có sẵn thông tin kết quả, tức là label tương ứng cho mỗi cột dữ liệu đầu vào. Tuy nhiên ở một số trường hợp ta sẽ không biết được những label này.

Mục đích của thuật toán này là phân cụm những dữ liệu sao cho ở mỗi cụm chúng có cùng tính chất.

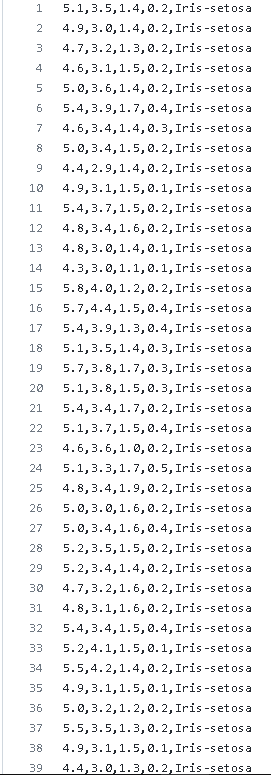
Ví dụ: Một công ty muốn tạo ra những chính sách ưu đãi cho các nhóm khách hàng khác nhau dựa trên sự tương tác của mỗi khách hàng với công ty. Giả sử công ty có rất nhiều dữ liệu của khách hàng như số tiền khách hàng mua sản phẩm của công ty, độ tuổi, thành phố, nghề nghiệp,… Bằng việc sử dụng thuật toán phân cụm ta sẽ có thể chia các nhóm khách hàng thành nhiều nhóm khác nhau để thuận tiện cho việc đưa ra các chiến lược sau này.

Để tính toán được một nhóm giống nhau từ bộ dữ liệu, ta sử dụng khoảng cách euclidean như một phép đo. Quá trình thực hiện thuật toán như sau:

Đầu tiên chúng ta sẽ tạo k điểm trung tâm, hay được biết là giá trị trung tâm của từng cụm một cách ngẫu nhiên. Tiếp đến, chúng ta phân loại những điểm dữ liệu từ bộ dữ liệu vào từng cụm bằng cách tính khoảng cách euclidean từ điểm đó đến điểm trung tâm, đồng thời ta sẽ cập nhập lại điểm trung tâm. Quá trình này sẽ được lặp cho đến khi có kết quả tối ưu nhất.

Để minh họa rõ ràng hơn ta sẽ bắt tay vào lập trình trên Python.

Đầu tiên, ta vẫn sẽ đọc bộ dữ liệu, giả sử đây là dữ liệu ta có.

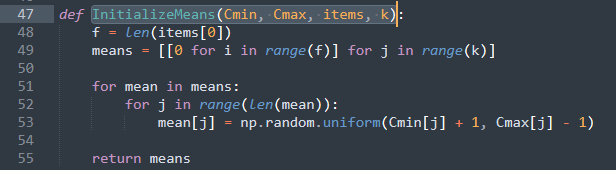
  

Đây là bộ dữ liệu loài hoa Iris tiêu chuẩn, ở đây ta sẽ loại bỏ cột cuối

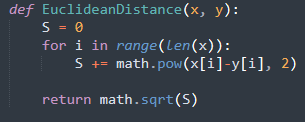
Bước kế tiếp ta sẽ tạo giá trị trung tâm của từng cụm, giá trị trung tâm thường là giá trị trung bình do đó bài toán mới có tên là K-means (trị trung bình). Thực chất ta chỉ cần tìm max và min của từng cột dữ liệu và dùng hàm random để tạo ra một giá trị ngẫu nhiên trong khoảng max, min ta tìm được.



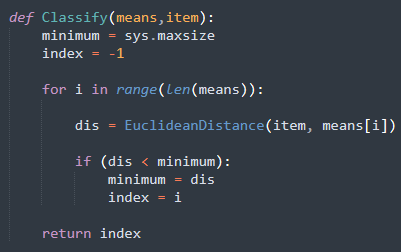
Như ta thấy ở bộ dữ liệu ta có 4 cột feature và đây là giá trị min và max tương ứng ở mỗi cột. Kế đến ta tạo giá trị trung tâm trong khoảng này. Giả sử bài toán này ta sẽ phân thành 3 cụm, tương ứng ta sẽ có 3 giá trị trung tâm, với mỗi giá trị có 4 cột feature.



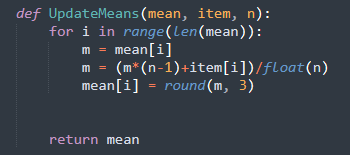


Vậy là ta đã có 3 giá trị trung tâm tương ứng ban đầu, kế tiếp ta tính khoảng cách Euclidean từ mỗi điểm dữ liệu tới các điểm trung tâm với công thức d(p,q) = 2. 

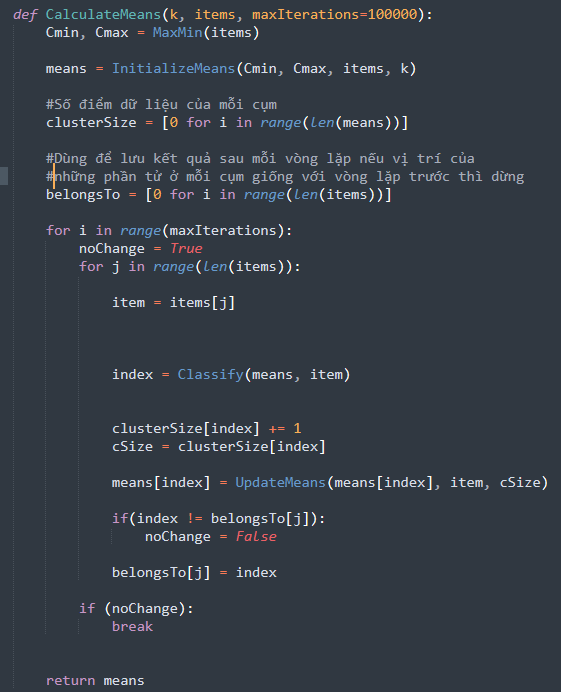
Và phân loại các điểm dữ liệu vào các cụm tương ứng dựa vào khoảng cách nhỏ nhất tới điểm đó. Đồng thời với mỗi lần 1 điểm được thêm vào cụm ta phải cập nhập lại giá trị trung bình của cụm đó.



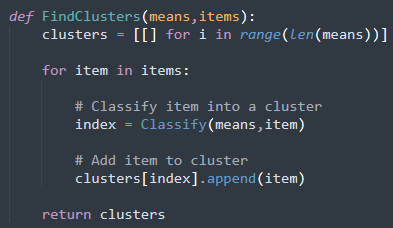
Ở đây hàm Classify chỉ trả về vị trị cụm mà một điểm dữ liệu thuộc về, chúng ta chưa thực hiện việc phân loại cụ thể.



Vậy là ta đã có đủ tất cả các hàm cần thiết, bây giờ ta tạo một hàm tính toán với số lần lặp ta qui định để cho ra giá trị trung tâm tối ưu nhất.



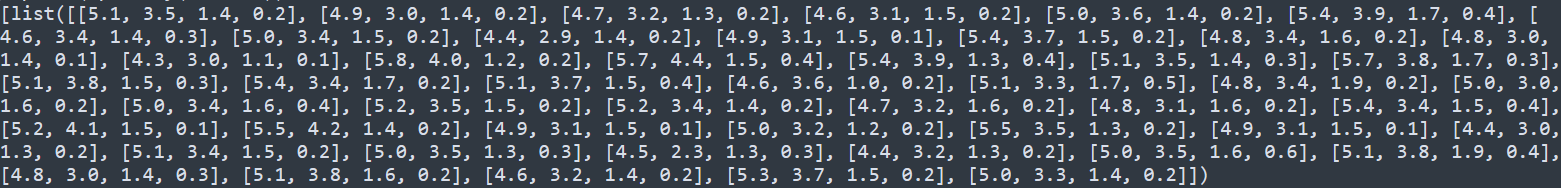
Kết quả trả về từ hàm CalculateMeans sẽ là giá trị trung tâm tối ưu nhất. Khi đã có được giá trị trung tâm, việc cuối cùng là phân loại từng điểm dữ liệu vào nó. Ta dùng hàm sau:



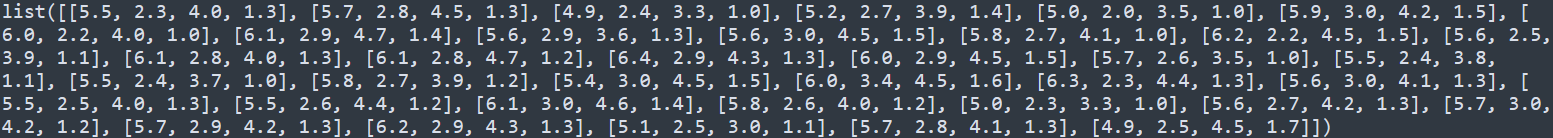
Khi ta có vị trí mà một điểm dữ liệu thuộc về, ta chỉ cần thêm nó vào một list. Như ta giả sử có 3 cụm, ta sẽ có 3 list được đánh vị trí lần lượt là cluster[0], cluster[1], cluster[2] và nếu điểm dữ liệu có vị trí 0,1 hoặc 2 thì sẽ được thêm vào tương ứng.

Đây là kết quả ta thu được từ bộ dữ liệu ban đầu:

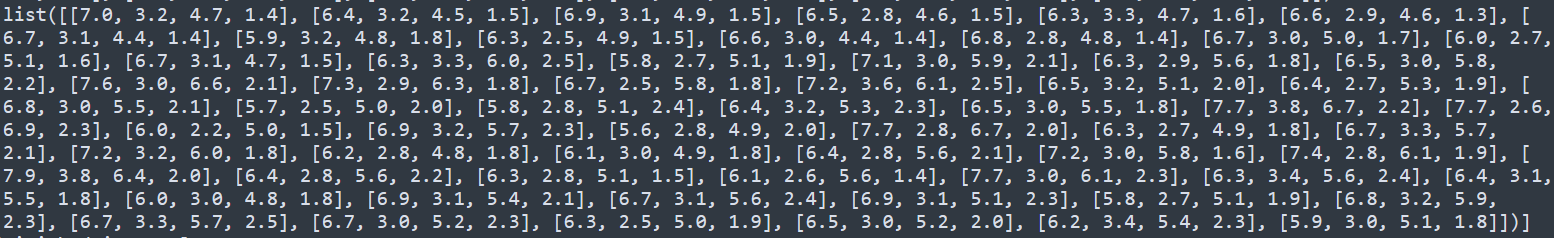
Cụm thứ nhất:

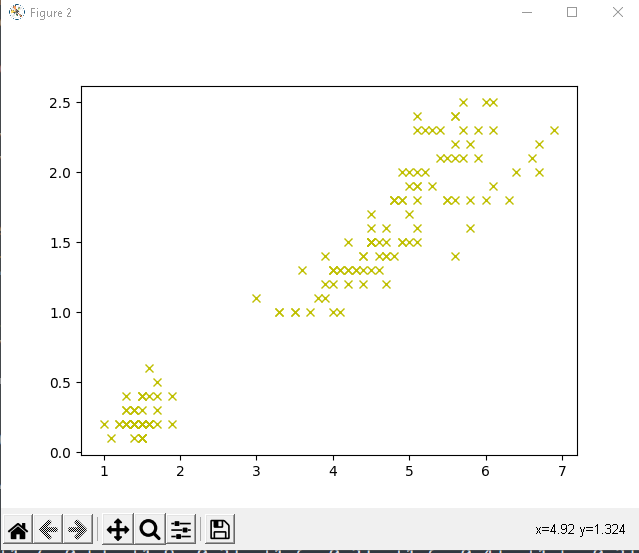
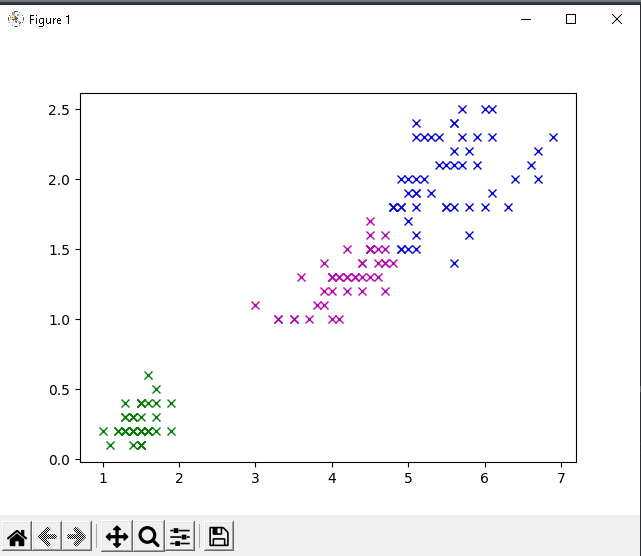


Cụm thứ hai:



Cụm thứ ba:



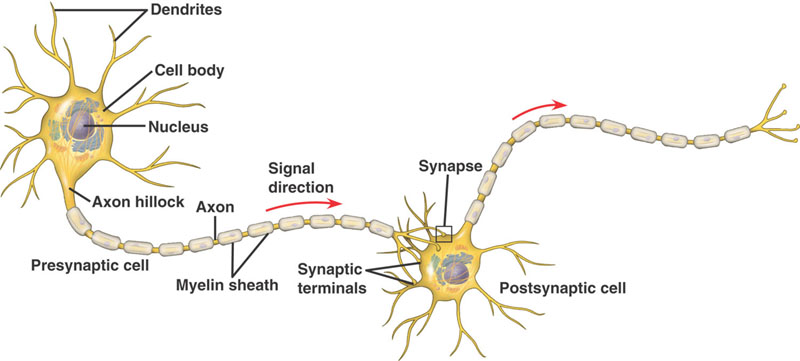
Dữ liệu ban đầu Dữ liệu sau khi phân loại

1. MẠNG THẦN KINH NHÂN TẠO – ATIFICIAL NEURAL NETWORKS
   1. Giới thiệu

Việc nghiên cứu mạng thần kinh nhân tạo (ANNs) được lấy cảm hứng bỏi một phần từ hệ thống học sinh học được xây dựng của những mạng lưới neuron được kết nối rất phức tạp ở trong não con người. Bộ não chứa một mạng lưới được kết nối dày đặc xấp xỉ 10^11 – 10^12 neurons, trung bình trên mỗi một neuron được kết nối sẽ có 10^4 – 10^5 neurol khác. Nhìn chung thì ANNs được xây dựng trên một bộ đơn vị đơn giản được kết nối dày đặc mà ở đó mỗi một đơn vị sẽ lấy một con số ngõ vào giá trị thực và tạo ra một giá trị thực ở ngõ ra.

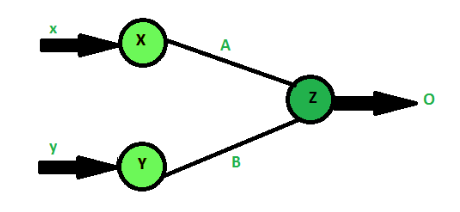
Thực chất mạng lưới thần kinh sinh học rất phức tạp do đó ANNs sẽ không mô hình hóa toàn bộ được. Một vài thông tin được đề cập trong bảng dưới đây:

|  |  |
| --- | --- |
| Neuron sinh học | Neuron nhân tạo |
| Thành phần chính: Axions, synap, sợi nhánh (dendrites). | Thành phần chính: Nodes, Input, Output, Weights, Bias. |
| Thông tin đến từ những neuron khác ở dạng xung điện từ sẽ đi vào sợi nhánh ở điểm kết nối gọi là synap. Thông tin sẽ theo dòng từ sợi nhanh để cell để xử lí. Tín hiệu ngõ ra, một chuỗi những xung sẽ được gửi xuống axon đến synap của neuron khác. | Sự bố trí và sự kết nối của những neuron tạo nên mạng lưới và có ba lớp. Lớp đầu tiên được gọi là lớp input và là lớp duy nhất giao tiếp với tín hiệu bên ngoài. Lớp input sẽ chuyển tín hiệu đến những neuron ở lớp tiếp theo chính là hidden layer. Hidden layer sẽ trích xuất những feature hay những mẫu có liên quan từ tín hiệu đã nhận. Những feature và mẫu đó có thể được cân nhắc là quan trọng hay không sau đó sẽ được chuyển trực tiếp đến output layer, lớp cuối cùng của mạng. |
| Một synap có thể tăng hoặc giảm cường độ kết nối. Đây là nơi mà thông tin được lưu trữ. | Những tín hiệu nhân tạo có thể được thay đổi bởi trọng số tương tự việc thay đổi vật lý xảy ra trong synap. |
| Xấp xỉ 10^11 neuron | 10^2 – 10^4 neurol với kĩ thuật hiện tại |



Hình 12 – Neuron sinh học

Vậy làm thế nào mà một neuron hoạt động? Giả sử chúng ta có hai neuron X và Y phát tín hiệu đến một neuron khác là Z. Vậy thì X và Y là neuron input cho việc phát tín hiệu và Z là neuron output cho việc nhận tín hiệu. Neuron input được kết nối đến neuron output qua các liên kết (A và B) được chỉ ra trong hình:



Hình 13 – Cấu trúc mạng neuron đơn giản

Đối với kiến trúc neuron ở trên thì đầu vào được tính toán bằng cách:

I = Ax + By

Ở đó x và y là activations của neuron input X và Y. Ngõ ra z của neuron output có thể đạt được bằng cách áp dụng activations lên đầu vào.

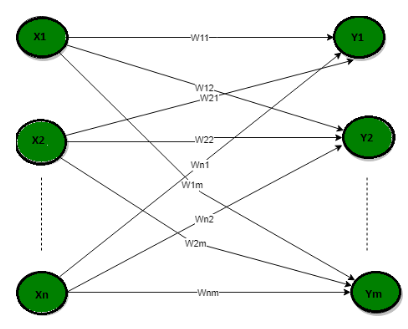
O = f(I)

Output = Function(Input đã được tính)

Function mà được áp dụng trên input được gọi là activation function. Có rất nhiều activation function sẵn có cho việc này.

Thông thường có năm loại kiến trúc kết nối neuron cơ bản:

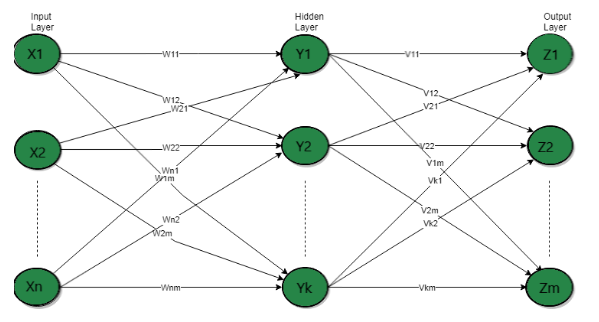
. Mạng chuyển tiếp một lớp



Hình 14 – Mạng chuyển tiếp một lớp

Trong loại kiến trúc này ta chỉ có hai lớp là input layer và output layer nhưng input layer là không được tính bởi vì không có tính toán nào diễn ra ở lớp này. Output layer được hình thành từ những trọng số (weights) khác nhau được áp dụng đến những nút input

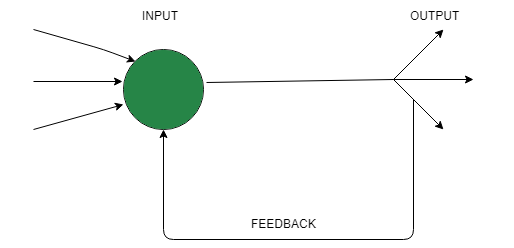
. Mạng chuyển tiếp nhiều lớp



Hình 15 – Mạng chuyển tiếp nhiều lớp

Trong kiến trúc này ta sẽ có thêm hidden layer, sự tồn tại của một hidden layer cho phép mạng tính toán mạnh hơn bởi vì thông tin đi qua hàm input và những tính toán trung gian được dùng để xác định ngõ ra Z.

. Một nút với phản hồi



Hình 16 – Mạng một nút phản hồi

Khi ngõ ra có được dẫn trực tiếp như ngõ vào của cùng một lớp hoặc những nút của lớp trước đó được gọi là mạng phản hồi. Mạng đệ quy là một mạng phản hồi với vòng lặp khép kín. Hình trên chỉ ra một mạng đơn đệ quy có một neuron với phản hồi của chính nó.

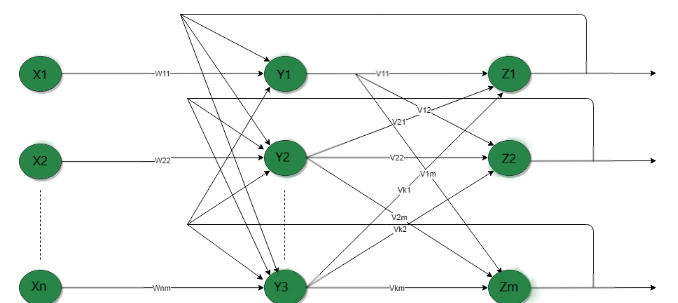
. Mạng đệ quy một lớp



Hình 17 – Mạng đệ qui một lớp

Mạng đệ quy một lớp như hình ta có thể thấy những kết nối phản hồi mà ở đó đầu ra của các nguyên tố xử lí được dẫn trực tiếp trở về chính nó hoặc đến những nguyên tố xử lí khác. Một mạng neuron đệ quy là một lớp của những mạng neuron nhân tạo nơi mà những kết nối giữa các nút tạo thành một đồ thị có hướng dọc theo một chuỗi. RNNs có thể sử dụng bộ nhớ nội để xử lí một chuỗi input.

. Mạng đệ qui đa lớp



Hình 18 – Mạng đệ qui đa lớp

Đầu ra nguyên tố xử lí có thể được dẫn trực tiếp tới nguyên tố xử lí trong cùng một lớp và trong lớp trước đó hình thành một mạng đệ quy đa lớp. Chúng thực hiện cùng một nhiệm vụ cho mỗi nguyên tố của một chuỗi, với đầu ra phụ thuộc vào việc tính toán trước đó.

Từ những thông tin trên ta có thể thấy những thành phần của một mạng neuron bao gồm:

Input layer: Lớp này sẽ chấp nhận các feature đầu vào, cung cấp thông tin từ thế giới bên ngoài vào mạng, không có tính toán xảy ra ở lớp này, chỉ đơn thuần chuyển thông tin tới hidden layer.

Hidden layer: Thực hiện một vài loại tính toán từ feature từ input layer và chuyển đổi kết quả tới output layer

Output layer: Lớp này sẽ mang thông tin được học bởi mạng ra thế giới bên ngoài.

Như ta đã biết việc chuyển từ đầu ra của một lớp đến một lớp khác thường sử dụng activation function. Activation function sẽ quyết định xem một neuron nên được kích hoạt hay không bằng cách tính toán tổng trọng số và cộng thêm bias với nó. Mục đính của activation function là khử sự tuyến tính trong ngõ ra của neuron.

Trong mạng neuron ta muốn cập nhập các trọng số, bias của neuron dựa theo lỗi tức sự sai lệch ở ngõ ra. Tiến trình này được gọi là back-propagation. Activation function sẽ cho phép back-propagation bởi vì gradient được cung cấp cùng với lỗi để cập nhập weights và bias. Bên cạnh đó nếu không có activation function thì mạng sẽ trở thành mạng mô hình linear regression thông thường, do đó việc khử tuyến tính sẽ giúp mạng thực hiện những nhiệm vụ phức tạp hơn.

Những biến thể của activation function:

1. Linear function

Linear function có biểu thức tương tự như một đường thẳng ví dụ: y=x

Không cần biết bao nhiêu lớp chúng ta có, nếu tất cả là tuyến tính từ đầu, thì activation của lớp cuối cùng không gì khác ngoài một linear function.

Linear function được sử dụng chỉ ở một nơi ví dụ như output layer.

1. Sigmoid function

A = 1/(1+ e-x)

Là một hàm phi tuyến tính.

Thường được sử dụng trong lớp đầu ra của loại phân loại nhị phân, trong đó kết quả là 0 hoặc 1.

1. Tanh function

f(x) = tanh(x) = 2/(1+e-2x) – 1

Là một hàm phi tuyến tính.

Thường được dùng trong hidden layer của một mạng neuron vì giá trị của nó nằm giữa -1 đến 1 vậy nên giá trị trung bình đầu ra là 0 hoặc rất gần với 0, do đó giúp trong việc căn giữa dữ liệu khiến việc học ở lớp tiếp theo dễ dàng hơn.

1. Relu function

A(x) = max(0,x), cho ngõ ra là x nếu x dương và 0 khi x âm.

Là một hàm phi tuyến tính.

Relu thì ít tính toán dài dòng hơn là tanh và sigmoid bởi vì nó liên quan đến các toán tử toán học đơn giản hơn. Ở một thời điểm chỉ một neuron được kích hoạt khiến việc tính toán dễ dàng.

1. Softmax function

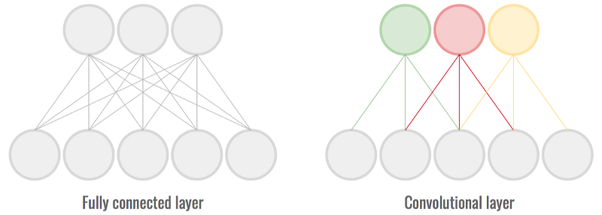
Giống với sigmoid function nhưng mà tiện lợi hơn khi xử lí với đa lớp.

Là một hàm phi tuyến tính.

Được dùng khi ta có nhiều hơn một lớp, softmax function thường được tìm thấy trong những vấn đề phân loại ảnh ở layer output. Đầu ra sẽ là xác suất của các lớp tương ứng.

* 1. Mạng neuron tích chập – Convolution Neural Network

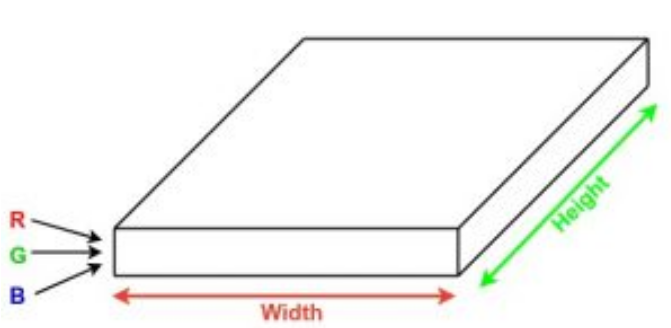
Mạng neuron tích chập ra đời giải quyết vấn đề liên quan tới dữ liệu ảnh. Như ta đã biết một mạng neuron thông thường nếu dữ liệu đầu vào là một ảnh màu sẽ có số lượng neuron rất lớn.



Hình 19 – Mạng tích chập với mạng neuron thông thường

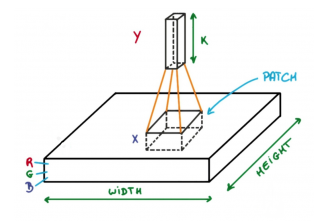
Khi số lượng neuron tăng thì các liên kết tức các trọng số theo đó cũng tăng lên dẫn đến việc tính toán sẽ rất phức tạp, khả năng xử lí dữ liệu sẽ rất lớn. Trong khi việc dùng lớp tích chập mỗi một neuron chỉ liên kết với neuron gần nhất của lớp tiếp theo.

Giả sử ta có một hình ảnh, có thể được biểu diễn dưới dạng hình khối có chiều dài, chiều rộng (kích thước của ảnh) và chiều cao tức là các màu để biểu diễn một ảnh (thường là đỏ, lục, lam).



Hình 20 – Cấu trúc của một dữ liệu ảnh

Bây giờ ta sẽ lấy một patch nhỏ của hình ảnh và chạy một mạng neuron nhỏ trên nó, giả sử có k đầu ra và biểu diễn chúng theo chiều dọc. Bây giờ hãy trượt mạng neuron đó trên toàn hình ảnh, kết quả ta sẽ nhận được một hình ảnh khác với chiều rộng, chiều cao và chiều sâu khác nhau. Thay vì R, G và B bây giờ ta sẽ có nhiều kênh màu hơn nhưng chiều rộng và chiều cao sẽ ngắn hơn. Quá trình này được gọi là tích chập, do đã giảm được kích thước ảnh cho nên ta sẽ có ít trọng số hơn.



Hình 21 – Minh họa tích chập

Toàn bộ tiến trình tích chập được miêu tả thuật toán như sau:

. Những lớp tích chập chứa một bộ những filter có khả năng học (một patch ở ảnh trên). Mỗi filter sẽ có chiều rộng và chiều cao nhỏ và cùng chiều sâu như dữ liệu đầu vào (3 nếu input layer là ảnh).

. Cho ví dụ, nếu chúng ta chạy tích chập trên một ảnh với kích thước 34 x 34 x 3. Kích cỡ có thể của filter có thể là a x a x 3, a có thể là 3, 5, 7,… Nhưng nhỏ hơn so với kích cỡ của ảnh.

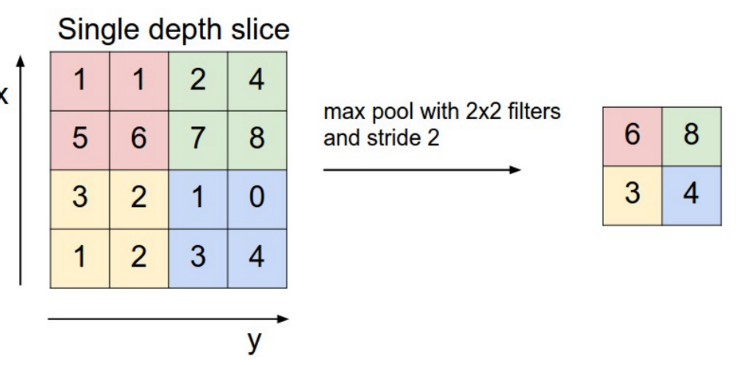
. Trong quá trình chuyển tiếp, ta trượt mỗi một filter qua toàn bộ dữ liệu đầu vào từng bước (step) một ở đó step được gọi là stride (giá trị có thể là 2, 3 hoặc thậm chí là 4 cho ảnh có kích thước lớn) và tính phép nhân ma trận giữa trọng số của filter và patch của dữ liệu đầu vào.

. Vì ta trượt những filter, ta sẽ nhận được đầu ra 2-D cho mỗi filter và ta sẽ xếp chúng chồng lên nhau, kết quả ta có dữ liệu đầu ra có độ sâu bằng với số lượng filter. Mạng sẽ học tất cả những filter.

Cách dùng các lớp để xây dựng mạng neuron tích chập:

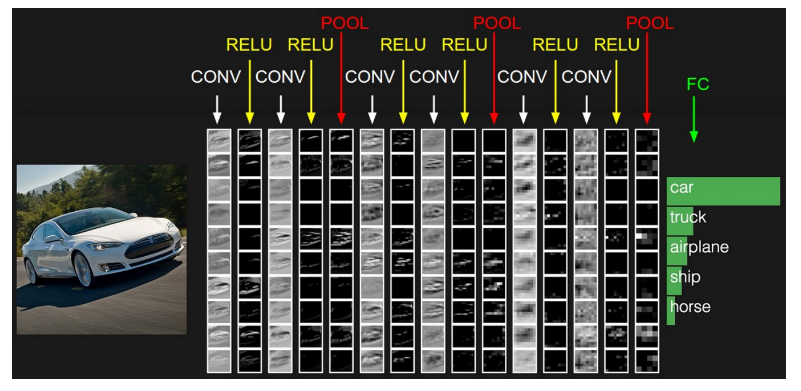
Giả sử ảnh đầu vào của ta có kích thước 32 x 32 x 3.

1. Input layer: Lớp này sẽ giữ thô ảnh với độ rộng 32, cao 32 và độ sâu 3.
2. Convolution layer: Lớp này sẽ tích toán giá trị đầu ra bằng cách thực hiện phép nhân ma trận giữa tất cả filter và patch của ảnh. Giả sử ta dùng tổng cộng là 12 filters cho lớp này chúng ta sẽ nhận dữ liệu đầu ra với kích thước 32 x 32 x 12.
3. Activation function layer: Lớp này sẽ áp dụng một hàm activation phù hợp với đầu ra của lớp tích chập. Một số activation function thông thường là RELU, Sigmoid, Tanh,…Dữ liệu với giữ nguyên kích thước 32 x 32 x 12.
4. Pool layer: Lớp này được thêm vào định kì trong mạng tích chập và chức năng chính của nó là giảm kích cỡ của dữ liệu khiến việc tính toán nhanh và ngăn chặn overfitting. Hai loại pooling layer thường dùng là max pooling và average pooling. Nếu ta dùng max pooling 2 x 2 filter và stride 2, dữ liệu kết quả sẽ có kích thước 16 x 16 x 12.



Hình 22 - Max pooling

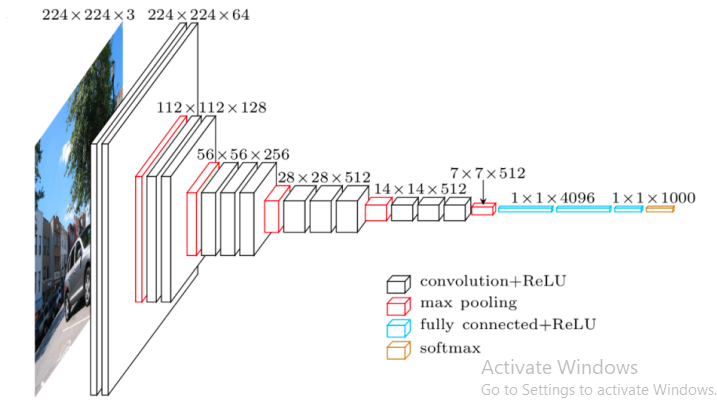
1. Fully connected layer: Lớp này là một lớp mạng neuron thông thường, lấy input từ các lớp trước và tính toán thông số và đầu ra của class, mảng một chiều sẽ bằng với số lượng của class.



Hình 23 – Minh họa một mô hình đầy đủ

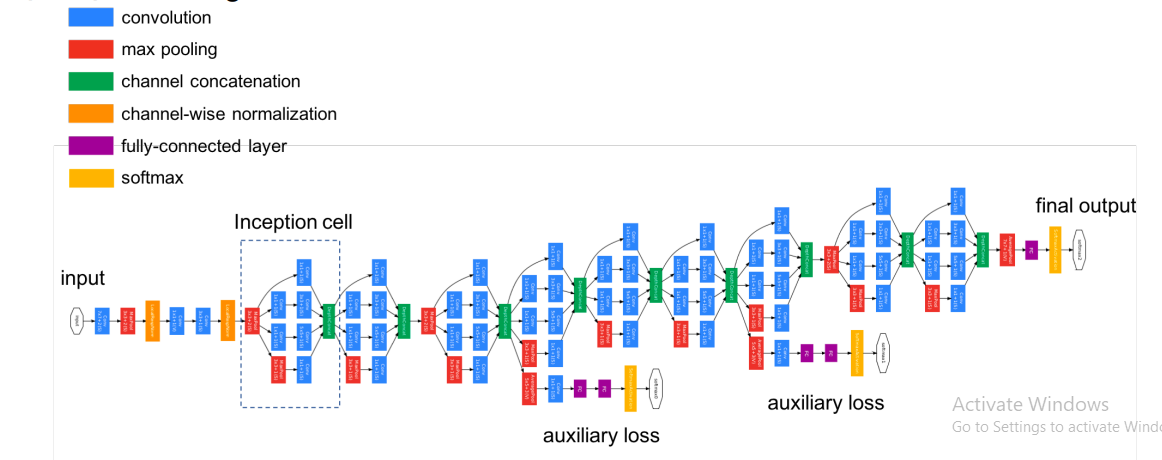
Một số mạng CNN nổi tiếng:

VGG16:



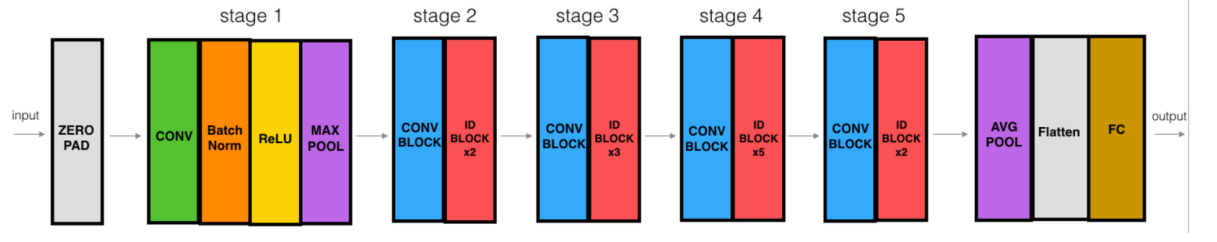
Hình 24 – VGG16

Inception:



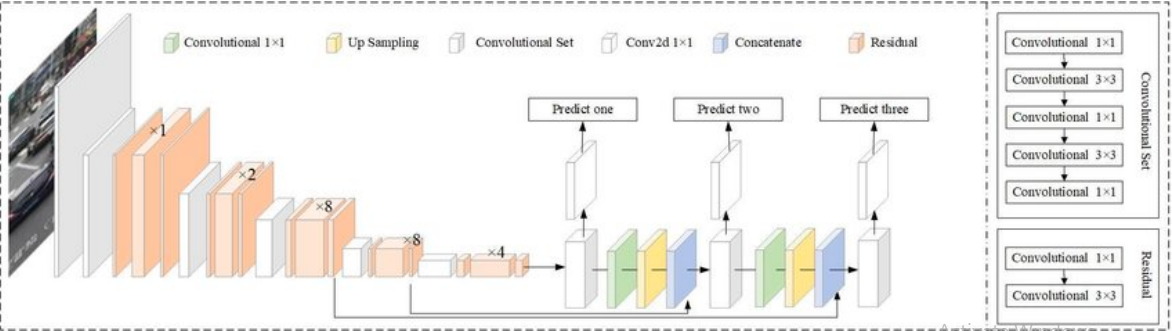
Hình 25 - Inception

ResNet:



Hình 26 - ResNet

Darknet-53:



Hình 27 – Darknet - 53

* 1. Bài toán nhận diện cử chỉ động của tay

Để làm được bài toán này trước tiên ta cần tìm hiểu tổng quan một số thư viện và các thuật ngữ được sử dụng để thực hiện.

* + 1. Opencv

Là một thư viện đa nền tảng được dử dụng trong những ứng dụng computer vision. Tập trung chủ yếu trên việc xử lí ảnh, video capture và phân tích bao gồm cả việc nhận diện khuôn mặt hoặc nhận diện đối tượng.

* + 1. Tensorflow và Keras

Tensorflow là một thư viện mã nguồn mở hỗ trợ mạnh mẽ các phép toán học để tính toán trong machine learning và deep learning. Thực chất tensorflow được xây dựng dựa trên numpy ta đã tìm hiểu ở trước. Tensorflow được phát triển bởi Google.

Keras là một thư viện ‘high-level’ với phần ‘low-level’ là tensorflow. Keras có cú pháp đơn giản hơn tensorflow và có khả năng lựa chọn CPU hoặc GPU để chạy chương trình. Keras sẽ được sử dụng để phục vụ chủ yếu trong bài toán này.

* + - 1. Các khái niệm trong Keras

Sequential: Là nơi chứa các layer một sequential có thể có nhiều layer, tập hợp của các layer sẽ tạo nên một sequential ví dụ như việc sử dụng các lớp để xây dựng mạng tích chập ở mục 2.

Layers: Là các lớp chứa các neuron, có nhiều loại layer khác nhau như (lớp tích chập, lớp pooling, lớp activation,…)

Activation function: Là các hàm hay các biểu thức toán học được dùng ở mỗi neuron của một lớp.

Fit: Thực hiện việc train cho một model.

Loss: Là các hàm tổn thất.

Optimizer: Là các thuật toán để tìm ra các weights tối ưu (thường dùng Adam và SGD).

Evaluation: Là hàm dùng đánh giá model của chúng ta với một lượng nhỏ dữ liệu.

* + - 1. Xây dựng bài toán nhận diện cử chỉ

Các bước xây dựng bài toán:

. Tạo bộ dữ liệu

. Xây dựng model

. Compile model

. Train model

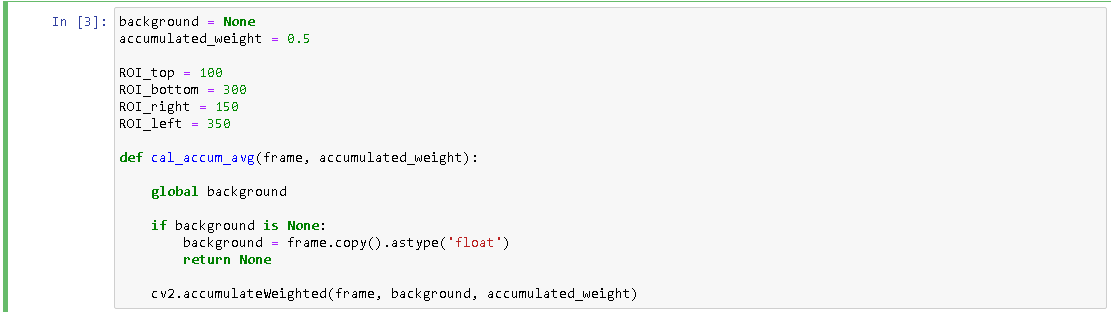
. Đánh giá model

. Đưa ra dự đoán

1. Tạo dữ liệu

Để có được dữ liệu ta có nhiều cách, ở đây ta sẽ tự tạo dữ liệu bằng cách chụp liên tục các khung hình từ camera của máy tính. Để làm được điều đó ta sẽ dùng thư viện opencv để mở camera của máy tính và lưu từng khung hình vào folder được tạo sẵn trong máy.

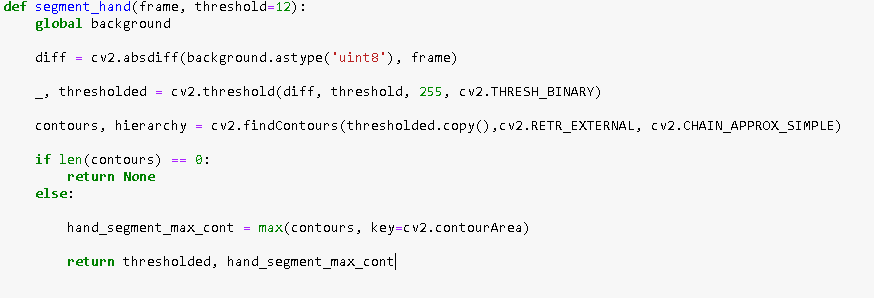
Như ta đã biết thì việc đưa bàn tay trước camera sẽ bao gồm cả background phía sau. Để việc nhận diện được tốt nhất ta sẽ dùng thuật toán background subtraction để loại bỏ phần background phía sau khi ta đưa tay vào trước camera. Ta sẽ tạo một hàm là cal\_accum\_avg. Thuật toán này có thể hiểu đơn giản ta sẽ tạo ra một khung hình chỉ với background ta muốn bỏ đi và thực hiện việc trừ từ khung hình có đối tượng ở phía trước background.



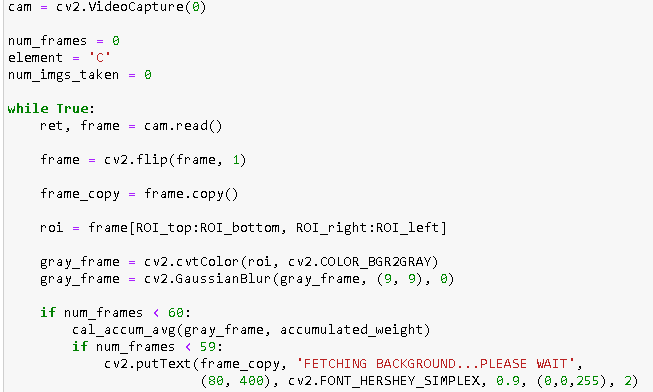
Đồng thời ta cũng khởi tạo ROI (Region of interest) chính là một ô vuông nhỏ để ta đặt tay vào đó cho máy nhận diện.

Tiếp đến để việc nhận dạng tối ưu hơn ta thiết lập threshold value, threshold value có thể hiểu như sau:

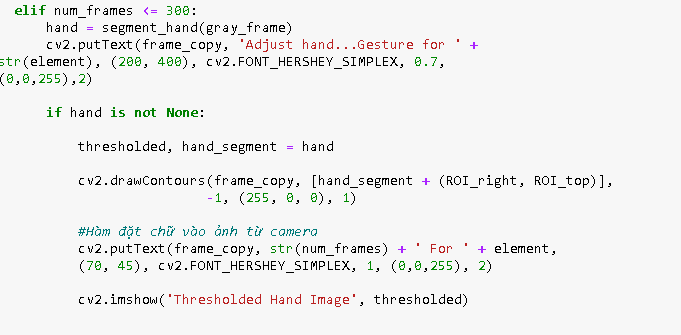
Giả sử ta có một ảnh đã được chuyển đổi sang ảnh xám (thường dùng cho việc xử lí dữ liệu) ta thiết lập một giá trị threshold và khi một điểm ảnh trên ảnh có giá trị lớn hơn giá trị threshold thì điểm ảnh đó sẽ là màu trắng, nhỏ hơn giá trị threshold thì sẽ là màu đen. Với ý tưởng như vậy khi ta đưa một vật vào nền background thì sẽ chỉ thấy vật đó có màu trắng và xung quanh chỉ là màu đen. Tất nhiên để xác định được rằng ta đã đưa đối tượng vô background hay chưa, ta cần một hàm hỗ trợ đó là cv2.findContours để tạo ra một tập điểm bao xung quanh đối tượng hình thành nên một đường cong, thực chất việc tìm contour là tìm đối tượng có màu trắng trên nền đen. Khi đã thực sự có đối tượng trước background thì hàm sẽ trả về giá trị (ở đây ta lấy giá trị max, tức là tập nhiều điểm nhất).

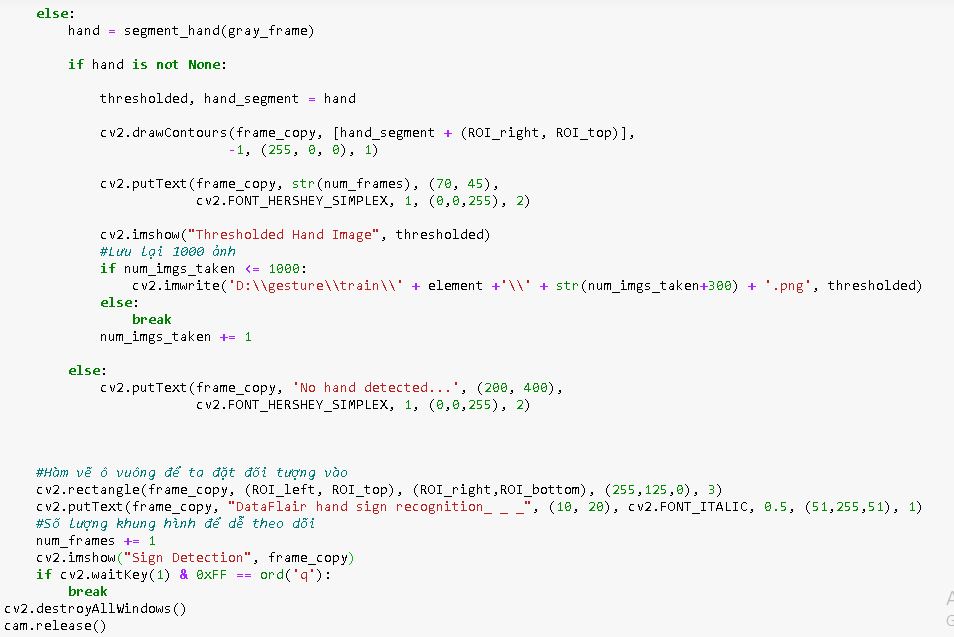


Sau khi đã thiêt lập các hàm cần thiết ta sẽ sử dụng camera của máy tính để bắt từng khung hình và lưu các dữ liệu là các cử chỉ tay mà ta muốn máy nhận diện. Ta sẽ chuyển đổi khung hình màu bắt được từ camera của máy thành ảnh xám. Trong 60 khung hình đầu tiên ta sẽ chỉ để background để tạo ra một mô hình background ta muốn trừ đi từ những khung hình sau. Từ khung hình 60 đến 300 ta chừa ra để xem có bất kì đối tượng nào được đặt vô background hay chưa. Và cuối cùng từ 300 khung hình trở lên ta chỉ đơn giản là lưu tất cả những ảnh (dữ liệu) của đối tượng mà ta đã đặt vào background ra một thư mục bất kì trong máy.





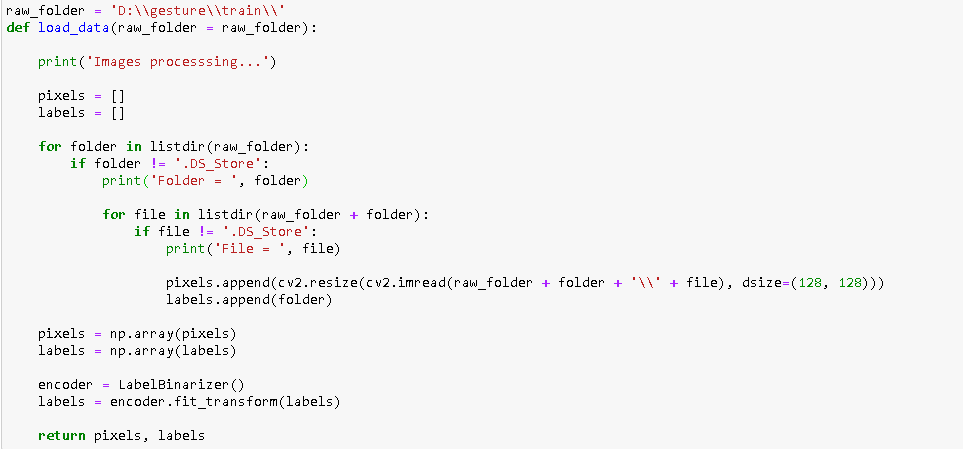




Sau khi ta chạy đoạn chương trình trên ta sẽ có 8 thư mục với tên lần lượt là ‘A’, ‘B’, ‘C’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’.

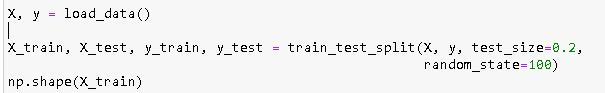


Khi đã có dữ liệu bước tiếp theo ta sẽ xử lí ảnh bằng cách resize lại các file ảnh và lưu vào một list, các nhãn tức các thư mục lưu vào một list khác.



Lưu ý là ta sẽ chuyển các thư mục thành dạng Onehot vector để tiện cho việc dự đoán.

Bước tiếp theo là bước phân chia dữ liệu thành tập test và tập train, ở phần trước ta đã quen thuộc với việc phân chia.



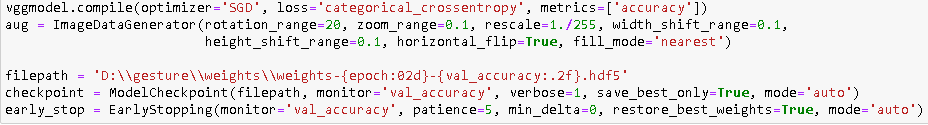
1. Xây dựng model

Ta sẽ dùng model VGG16 để việc train tối ưu hơn, vì đây là model đã được chuẩn hóa nên các weight của nó cho ra số liệu rất tốt, nhiệm vụ của ta ở đây chỉ điều chỉnh ở phần đuôi của VGG16, tức là phần tích chập của VGG16 ta sẽ giữ nguyên.



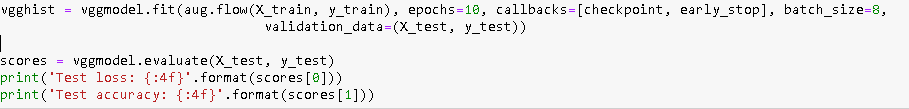
1. Compile model

Ở đây ta sẽ dùng thêm augment dữ liệu để cho việc train được linh hoạt hơn, bên cạnh đó ta kết hợp hai kĩ thuật callbacks là CheckPoint và EarlyStopping để lưu weights tối ưu nhất và dừng vòng lặp sớm nếu weights không cải thiện được nữa.



1. & 5) Train model và đánh giá

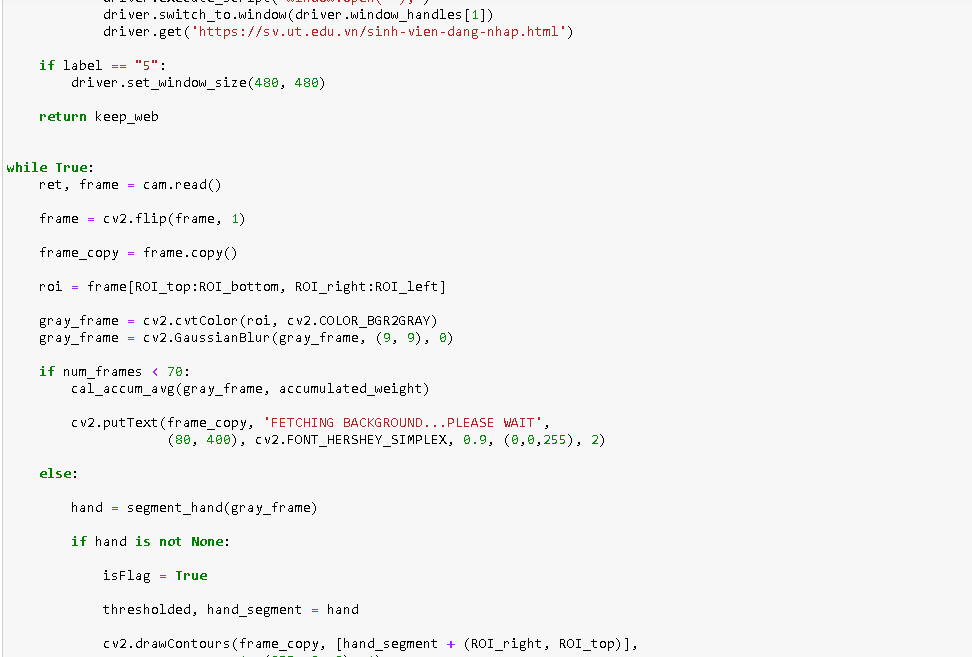
Ta tạo 10 vòng lặp với batch\_size là 8 và cho validation data là test set để chống việc dữ liệu quá overfit với train set. Ta cũng đồng thời đánh giá lại một lần nữa với bộ test để xem độ chính xác.

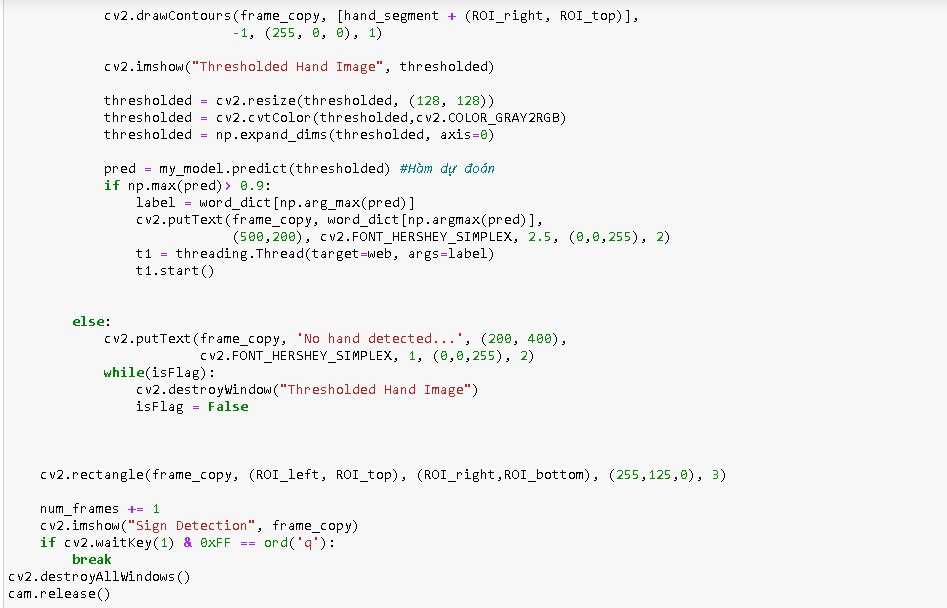


1. Đưa ra dự đoán

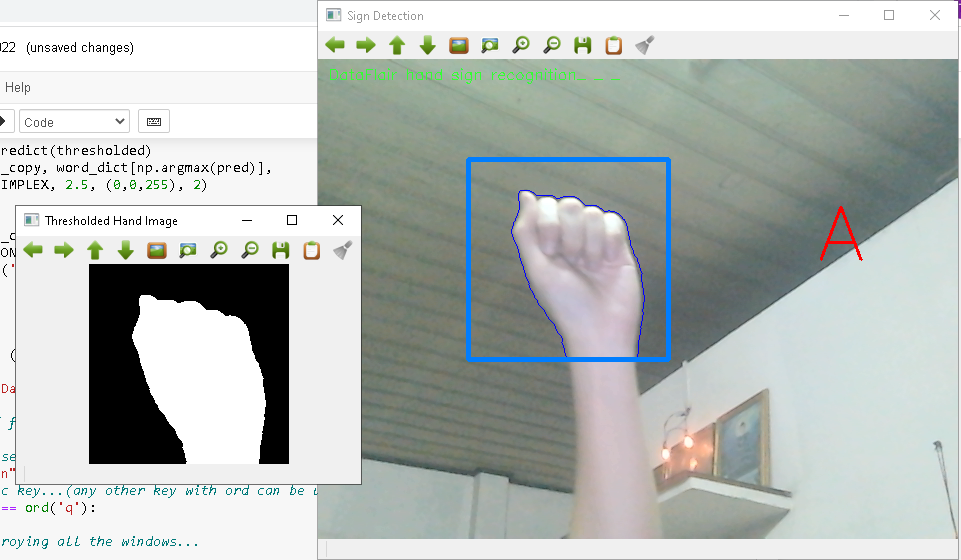
Ta sẽ tạo chương trình giống như lúc tạo data nhưng lần này ta sẽ không lưu ảnh mà thay vào đó là gọi hàm dự đoán của model và đưa lên màn hình, thêm vào một hàm web ở đó ta sẽ tương tác với trang web.

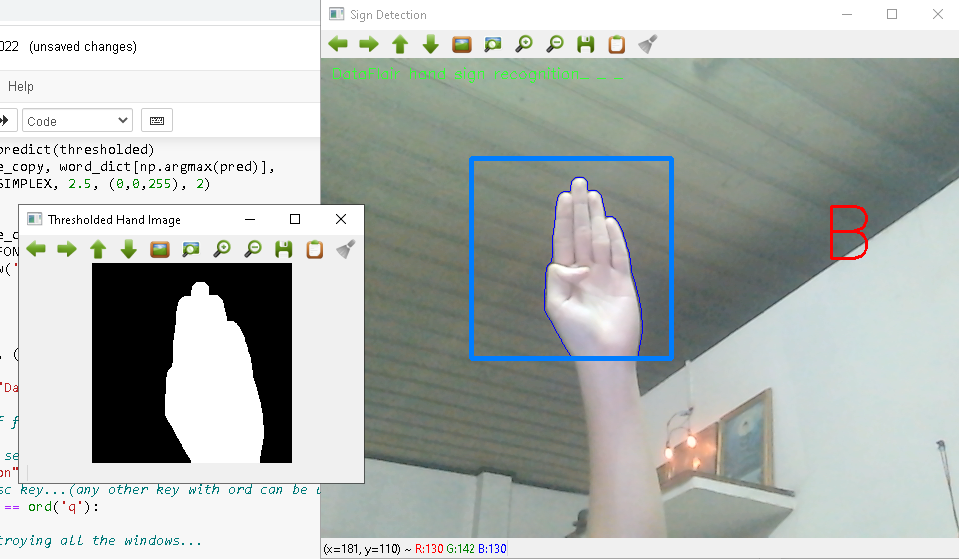


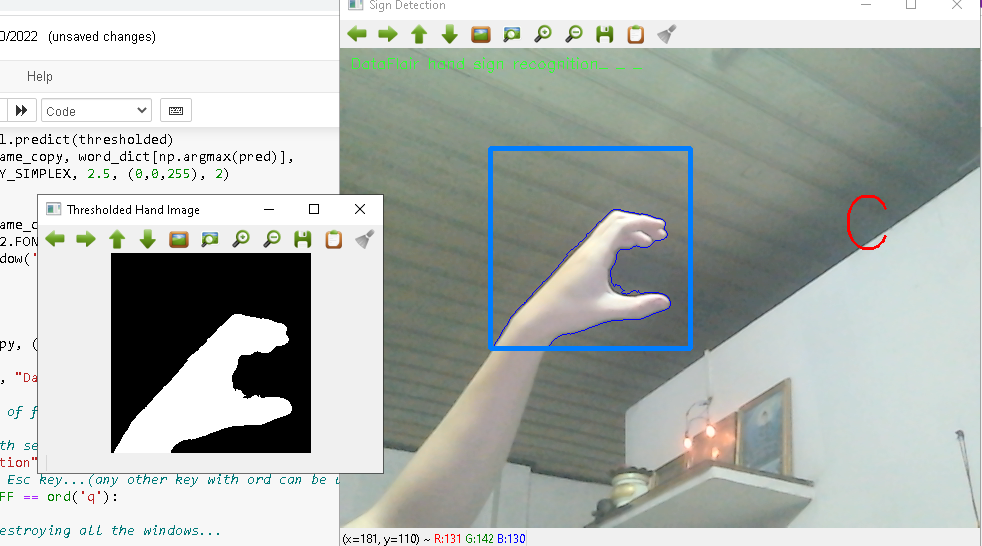


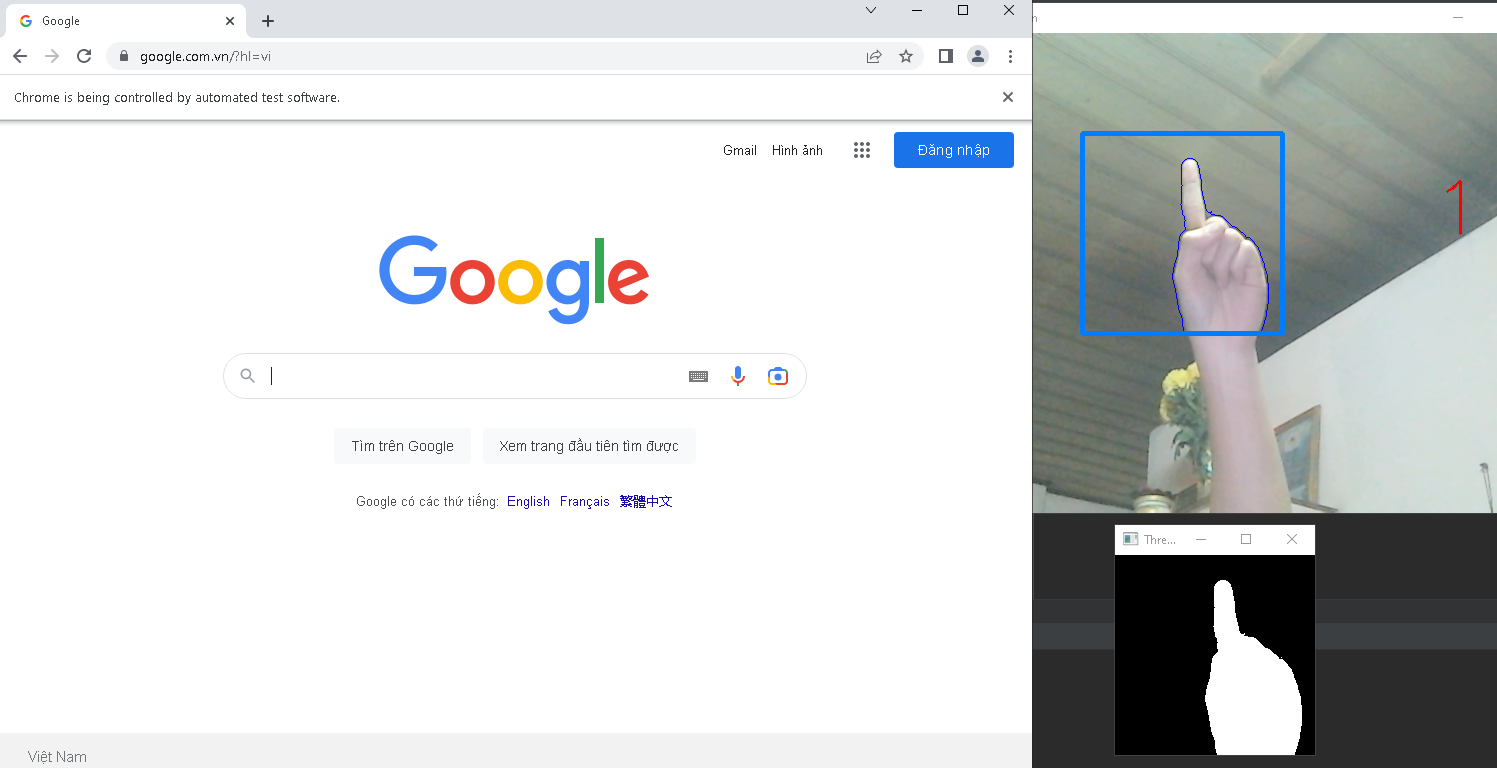


Kết quả ta sẽ được như hình mẫu bên dưới và bên cạnh đó ta quy định thêm một số những tác vụ với từng cử chỉ tay chẳng hạn như đóng mở một trang web bất kì với kí hiệu số 1, mở một tab mới với kí hiệu số 4, thu nhỏ cửa sổ với kí hiệu số 5.

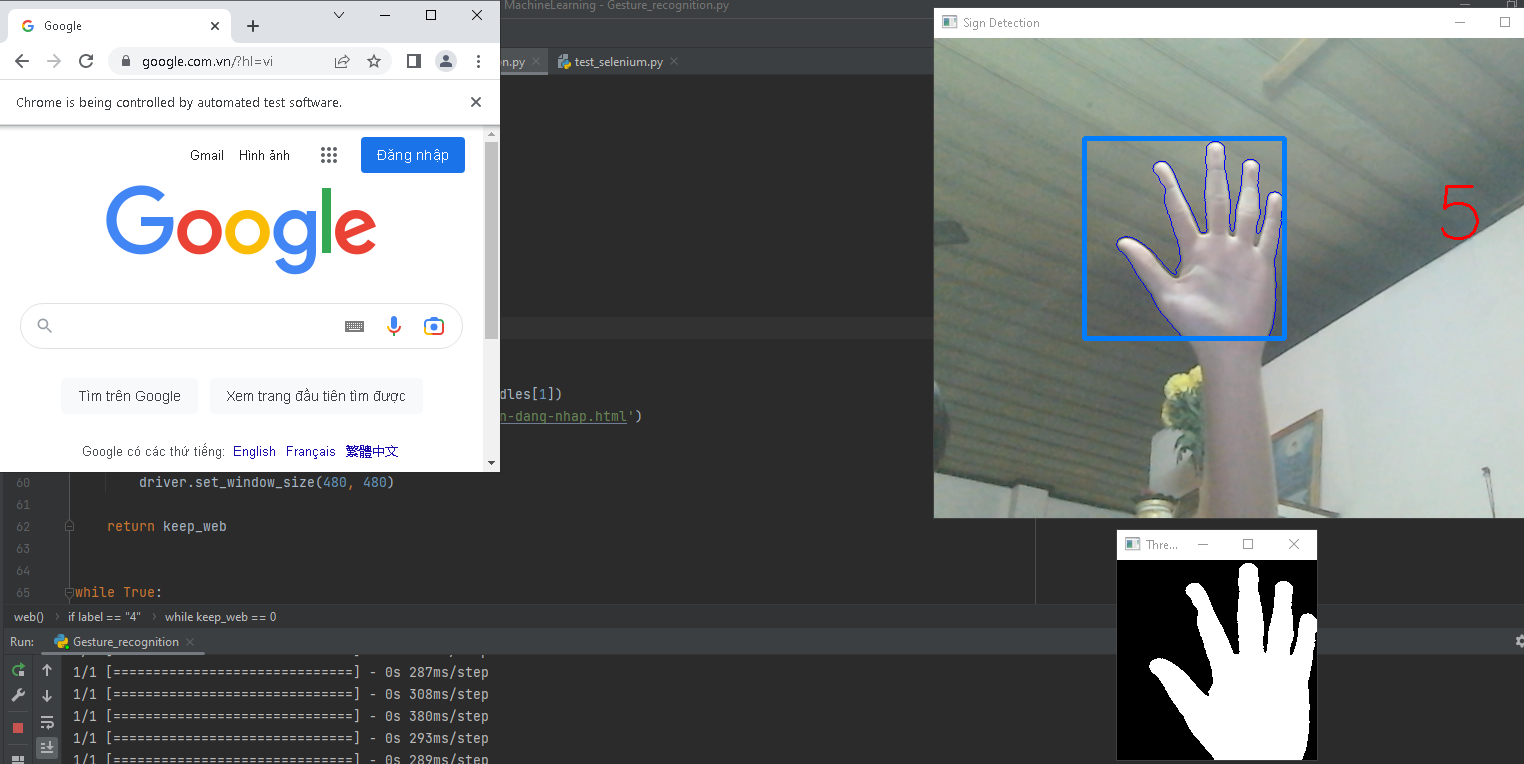












1. KẾT LUẬN
   1. Một số điểm hạn chế của luận văn

Luận văn vẫn còn chưa trình bày bao quát hết tất cả các thuật ngữ cũng như một số nội dung liên quan, cụ thể là:

. Các thuật toán machine learning chưa được khai thác hết.

. Việc liên kết giữa cơ sở dữ liệu để tạo dữ liệu cho machine learning chưa được triển khai.

. Thiếu trang thiết bị để có thể tạo ra một mô hình nhận dạng đa dạng cũng như là tính linh hoạt và chính xác cho deep learning với bài toán nhận dạng cử chỉ.

. Các thuật toán và các phương tiện được sử dụng trong model của deep learning chưa được đào sâu.

* 1. Hướng phát triển của luận văn

Từ những hạn chế được nêu ra ở trên ta có thể đưa ra một vài hướng phát triển cho luận văn như kết hợp thiết kế web để đẩy model nhận dạng lên phía backend và sử dụng các cơ sở dữ liệu. Tìm hiểu thêm cách thiết kế model để cho độ chính xác được cao nhất có thể. Khai thác thêm các thư viện được dùng trong xử lí ảnh, các kĩ thuật xử lí dữ liệu cũng như là việc kết hợp các thuật toán machine learning. Phát triển thêm model nhận dạng bằng cách thêm các cử chỉ và cách tương tác với model nhận diện hoặc nếu có thêm thời gian sẽ tạo ra mô hình nhận diện các hoạt động thông thường như đi bộ, chạy, nhảy,….

* 1. Kết luận

Việc nghiên cứu về AI nói chung và deep learning nói riêng là một chủ đề rất phức tạp và cần rất nhiều thời gian để có thể hiểu rõ, hiểu tường tận dòng dữ liệu mà máy tính xử lí cũng như là việc áp dụng các thuật toán sao cho khi dùng vào các ứng dụng trí tuệ nhân tạo cho khả năng vận hành tốt nhất. Tránh những tổn thất và lỗi là điều mà các kĩ sư hay các chuyên gia nhắm đến.

Qua việc thực luận văn về trí tuệ nhân tạo đã giúp cho bản thân em có thêm rất là nhiều kiến thức từ toán học nền tảng vốn đã được học ở năm nhất cho đến các kiến thức nâng cao để dùng vào lập trình, em còn được học thêm ngôn ngữ lập trình rất là mạnh và các thư viện cũng như là một số framework nổi tiếng. Được áp dụng những kiến thức tự tìm hiểu để xây dựng một mô hình deep learning. Tạo một nền tảng cơ bản cho em để tiếp tục học, tìm hiểu thêm các kiến thức liên quan sẵn sàng phục vụ cho cơ hội nghề nghiệp ở tương lai.

TÀI LIỆU THAM KHẢO

<https://www.khanacademy.org/math/linear-algebra>

<https://www.academia.edu/37074934/%C4%90%E1%BA%A0I_S%E1%BB%90_TUY%E1%BA%BEN_T%C3%8DNH_T%C3%A0i_li%E1%BB%87u_n%E1%BB%99i_b%E1%BB%99>

<https://www.dataversity.net/review-pros-cons-different-databases-relational-versus-non-relational/>

<https://en.wikipedia.org/wiki/Join_(SQL)>

<https://en.wikipedia.org/wiki/NoSQL>

<https://www.statology.org/tabular-data/>

<https://en.wikipedia.org/wiki/Extract,_transform,_load>

<https://www.json.org/json-en.html>

<https://en.wikipedia.org/wiki/XML>

<https://en.wikipedia.org/wiki/Regular_expression>

<https://en.wikipedia.org/wiki/Regular_expression>

<https://www.guru99.com/python-tutorials.html>

<https://www.geeksforgeeks.org/expressions-in-python/>

<https://howkteam.vn/course/lap-trinh-python-co-ban-37>

<https://www.w3schools.com/python/>

<https://www.guru99.com/numpy-tutorial.html>

<https://www.w3schools.com/python/numpy/default.asp>

<https://howkteam.vn/course/machine-learning-co-ban-voi-numpy-64>

<https://www.guru99.com/machine-learning-tutorial.html>

<https://machinelearningcoban.com/2016/12/26/introduce/>

<https://www.geeksforgeeks.org/machine-learning/>

<https://www.w3schools.com/python/python_ml_getting_started.asp>

<https://miai.vn/2020/04/21/nhan-dang-tien-viet-nam-voi-transfer-learning-vgg16-cnn-classify/>

<https://www.geeksforgeeks.org/dropout-in-neural-networks/>

<https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/>

<https://www.geeksforgeeks.org/opencv-python-tutorial/>

<https://www.geeksforgeeks.org/background-subtraction-in-an-image-using-concept-of-running-average/>

<https://www.geeksforgeeks.org/intuition-of-adam-optimizer/>

<https://www.geeksforgeeks.org/ml-common-loss-functions/>

<https://www.youtube.com/watch?v=TEHnfsqpUPc>

<https://www.youtube.com/watch?v=TY43q0w1Eh0&list=PLZPCoTKpEddAay-lItE-pn27uNNrApORH>