

# Basic Python - If-Else

*Hoàng-Nguyễn Vũ*

## 1 Cấu trúc if-else

Cấu trúc **if-else** là một cấu trúc điều kiện trong lập trình, được sử dụng để kiểm tra một điều kiện logic và thực hiện các đoạn mã khác nhau dựa trên kết quả của điều kiện đó (đúng hoặc sai).

- **if:** Kiểm tra điều kiện. Nếu điều kiện đúng, thực thi khối lệnh bên trong.
- **else:** Được thực thi khi điều kiện của **if** là sai.
- **elif (nếu có):** Kiểm tra điều kiện bổ sung khi điều kiện ban đầu không đúng.

### Cú pháp cơ bản trong Python:

```
1 if điều_kiện:  
2     # Lệnh được thực thi khi điều kiện đúng  
3 elif điều_kiện_khác:  
4     # Lệnh được thực thi nếu điều kiện khác đúng  
5 else:  
6     # Lệnh được thực thi nếu tất cả các điều kiện đều sai
```

### Ví dụ:

```
1 x = 10  
2 if x > 0:  
3     print("x là số dương")  
4 elif x == 0:  
5     print("x bằng 0")  
6 else:  
7     print("x là số âm")
```

## 2 Bài tập

### 2.1 Tính lịch Can Chi - làm quen với câu lệnh If/Else



- Can Chi là một hệ thống tính toán giờ, ngày, tháng, năm âm lịch của người trung quốc cổ đại. Can Chi có 10 thiên can và 12 địa chi. Tên gọi 10 thiên can Canh, Tân, Nhâm, Quý, Giáp, Ất, Bính, Đinh, Mậu, Kỷ. Tên gọi 12 địa chi Thân, Dậu, Tuất, Hợi, Tý, Sửu, Dần, Mão, Thìn, Ty, Ngọ, Mùi.
- Để tính được can chi, chúng ta dựa vào quy tắc sau đây:

– **Can:** lấy năm sinh chia cho 10 và lấy phần dư. Nếu phần dư bằng 0 tương ứng với Canh, 1 tương ứng với Tân, tiếp tục cho tới 9 tương ứng với Kỷ

Phần dư	0	1	2	3	4	5	6	7	8	9
Can	Canh	Tân	Nhâm	Quý	Giáp	Ất	Bính	Đinh	Mậu	Kỷ

– **Chi:** lấy năm sinh chia cho 12 và lấy phần dư. Nếu phần dư bằng 0 tương ứng với Thân, 1 tương ứng với Dậu, tiếp tục cho tới 11 tương ứng với Mùi

Phần dư	0	1	2	3	4	5	6	7	8	9	10	11
Chi	Thân	Dậu	Tuất	Hợi	Tý	Sửu	Dần	Mẹo	Thìn	Ty	Ngọ	Mùi

```

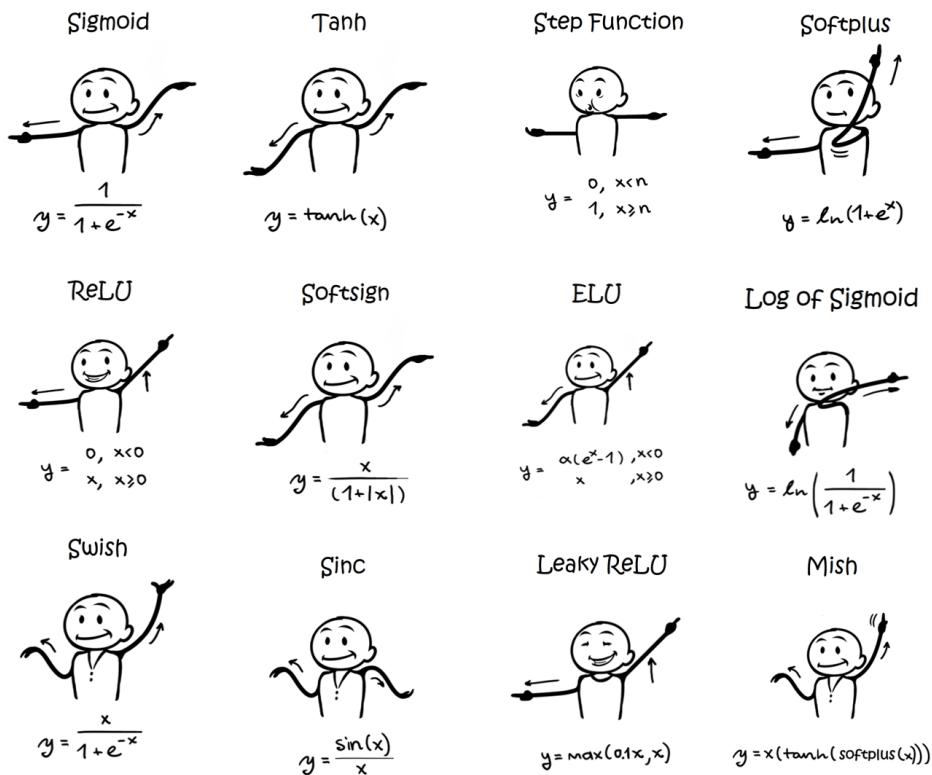
1 def calculate_can_chi_calendar(year):
2     result = ''
3     ##### Your code here #####
4     return result

```

#### Ví dụ:

- Test case 1: calculate\_can\_chi\_calendar(2025) → Ất Ty
- Test case 2: calculate\_can\_chi\_calendar(2024) → Giáp Thìn
- Test case 3: calculate\_can\_chi\_calendar(1997) → Dinh Sửu

## 2.2 Hàm kích hoạt - Activate function



**Activate Function** thường được sử dụng trong các mô hình học máy để áp dụng một hàm kích hoạt lên đầu ra của một tầng. Nó chuyển đổi dữ liệu đầu vào thành một không gian mong muốn hoặc không gian phi tuyến, giúp mô hình có thể học được các mẫu phức tạp. Một số hàm kích hoạt phổ biến bao gồm: *ReLU*, *Sigmoid*, và *Tanh*, mỗi hàm phù hợp với các nhiệm vụ và kiến trúc khác nhau.

### 1. ReLU (Rectified Linear Unit)

$$\text{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

```

1 def relu(x):
2     ##### Your code here #####

```

#### Examples:

- Test case 1:  $\text{relu}(3) \rightarrow 3$
- Test case 2:  $\text{relu}(-2) \rightarrow 0$
- Test case 3:  $\text{relu}(0) \rightarrow 0$

## 2. Leaky ReLU

$$\text{Leaky ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$$

```
1 def leaky_relu(x, alpha=0.01):
2     ##### Your code here #####
```

**Examples:**

- Test case 1:  $\text{leaky\_relu}(3) \rightarrow 3$
- Test case 2:  $\text{leaky\_relu}(-2) \rightarrow -0.02$
- Test case 3:  $\text{leaky\_relu}(0) \rightarrow 0$

## 3. Sigmoid

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

```
1 import math
2
3 def sigmoid(x):
4     ##### Your code here #####
```

**Examples:**

- Test case 1:  $\text{sigmoid}(0) \rightarrow 0.5$
- Test case 2:  $\text{sigmoid}(2) \rightarrow 0.8808$
- Test case 3:  $\text{sigmoid}(-2) \rightarrow 0.1192$

## 4. Tanh

$$\text{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

```
1 import math
2
3 def tanh(x):
4     ##### Your code here #####
```

**Examples:**

- Test case 1:  $\text{tanh}(0) \rightarrow 0.0$
- Test case 2:  $\text{tanh}(2) \rightarrow 0.964$
- Test case 3:  $\text{tanh}(150) \rightarrow 1$

## 5. ELU (Exponential Linear Unit)

$$\text{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases}$$

```
1 import math
2
3 def elu(x, alpha=1.0):
4     ##### Your code here #####
```

### Examples:

- Test case 1:  $\text{elu}(3) \rightarrow 3$
- Test case 2:  $\text{elu}(-1) \rightarrow -0.6321$
- Test case 3:  $\text{elu}(0) \rightarrow 0$

# List and Tuple

*Hoàng-Nguyên Vũ*

## 1. Giới thiệu:

- Tuple là một kiểu dữ liệu cơ bản trong Python, được sử dụng để lưu trữ tập hợp các phần tử có thứ tự. Tuple có thể chứa bất kỳ kiểu dữ liệu nào, bao gồm số nguyên, chuỗi, số thập phân, danh sách con, v.v.

## 2. Mô tả:

Bảng 1: Sự Giống Nhau và Khác Nhau giữa Tuple và List

Đặc Điểm	Tuple	List
Đặc Điểm Cơ Bản	Tập hợp các phần tử không thay đổi được, đặt trong cặp dấu ngoặc đơn.	Tập hợp các phần tử có thể thay đổi được, đặt trong cặp dấu ngoặc vuông.
Thay Đổi Dữ Liệu	Không thể thay đổi (immutable). Không thể thêm, xóa hoặc thay đổi các phần tử.	Có thể thay đổi (mutable). Có thể thêm, xóa và thay đổi các phần tử.
Sử Dụng	Thích hợp để bảo vệ dữ liệu.	Thích hợp cho các cấu trúc dữ liệu có thể thay đổi.
Hiệu Suất	Trong một số trường hợp, tuple có thể nhanh hơn	List có sự linh hoạt cao hơn.

## 3. Bài tập: Khởi tạo Tuple và thao tác tìm kiếm, trích xuất thông tin trên Tuple đó.

- **Câu 1:** Tạo mới hai Tuple: `my_tuple1 = (2,3)`, `my_tuple2 = (3,6)` mỗi Tuple có 2 phần tử đại diện cho một vector trong không gian 2D.
- **Câu 2:** In ra kết quả của tổng và tích 2 vector trên.
- **Câu 3:** In ra kết quả của khoảng cách của hai vector trên theo công thức.  
Biết  $\text{distance}(P, Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2}$
- **Câu 4:** In ra vị trí của phần tử có giá trị là 3  
Sử dụng cú pháp: `my_tuple.index(values)` để trích xuất vị trí của giá trị cần tìm.

```

1 my_tuple1 = ()
2 my_tuple2 = ()
3 # Your code here

```

### Output:

- **Câu 2:** `Result_vector1=(5,6), Result_vector2=(9,18)`
- **Câu 3:**  $\sqrt{10}=3.1622776601683795$ .
- **Câu 4:** `index=(1, 0)`.

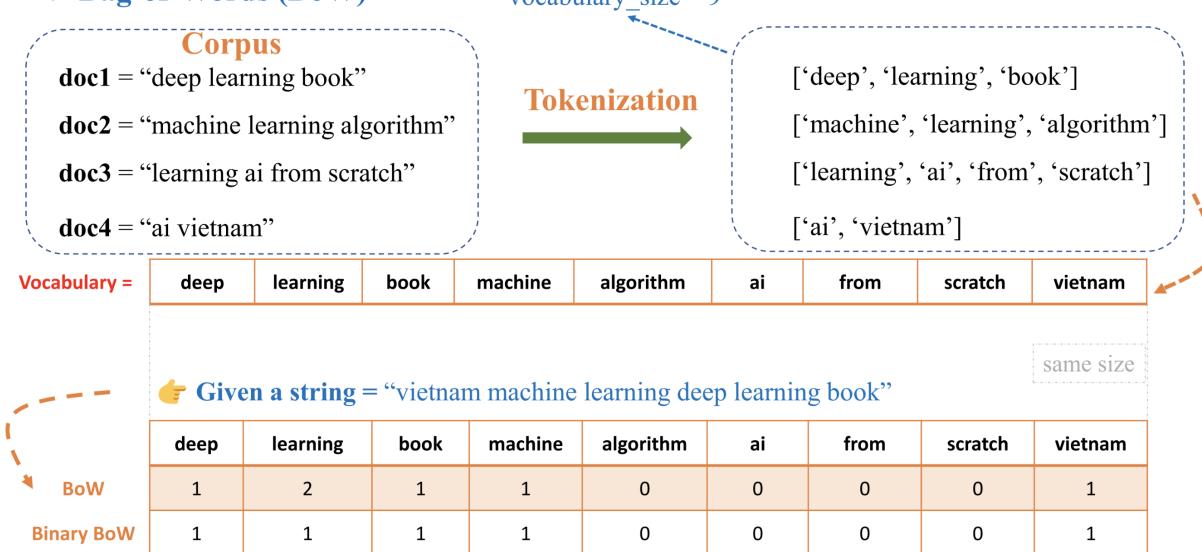
# Bag of Words (NLP)

*Hoàng-Nguyễn Vũ*

## 1. Mô tả:

- **Bag of Words** là một thuật toán hỗ trợ xử lý ngôn ngữ tự nhiên và mục đích của BoW là phân loại text hay văn bản. Ý tưởng của BoW là phân tích và phân nhóm dựa theo “Bag of Words” (corpus). Với test data mới, tiến hành tìm ra số lần từng từ của test data xuất hiện trong “Bag”. Cách thức thực hiện như sau:
  - **Bước 1:** Chia nhỏ văn bản thành các từ riêng lẻ.
  - **Bước 2:** Tạo một tập hợp các từ xuất hiện trong văn bản. Tập hợp này không có phần tử trùng nhau.
  - **Bước 3:** Biểu diễn văn bản input ở dạng vector: Mỗi câu (mỗi input) được biểu diễn bằng một vector, với mỗi phần tử trong vector thể hiện số lần xuất hiện của từ đó trong input.

### ❖ Bag-of-Words (BoW)



2. **Bài tập:** Tạo Bag-Of-Word cho tập dataset sau: *corpus* = ["Tôi thích môn Toán", "Tôi thích AI", "Tôi thích âm nhạc"]. Sau đó tạo list có tên *vector* để lưu vector sau khi thực hiện bước Tokenization đoạn văn bản sau: **Tôi thích AI thích Toán**, biết Bag-Of-Word được sắp theo thứ tự **tăng dần**

```
1 corpus = [ ''' Your Code Here ''']
2 # Your code here
```

#### Output:

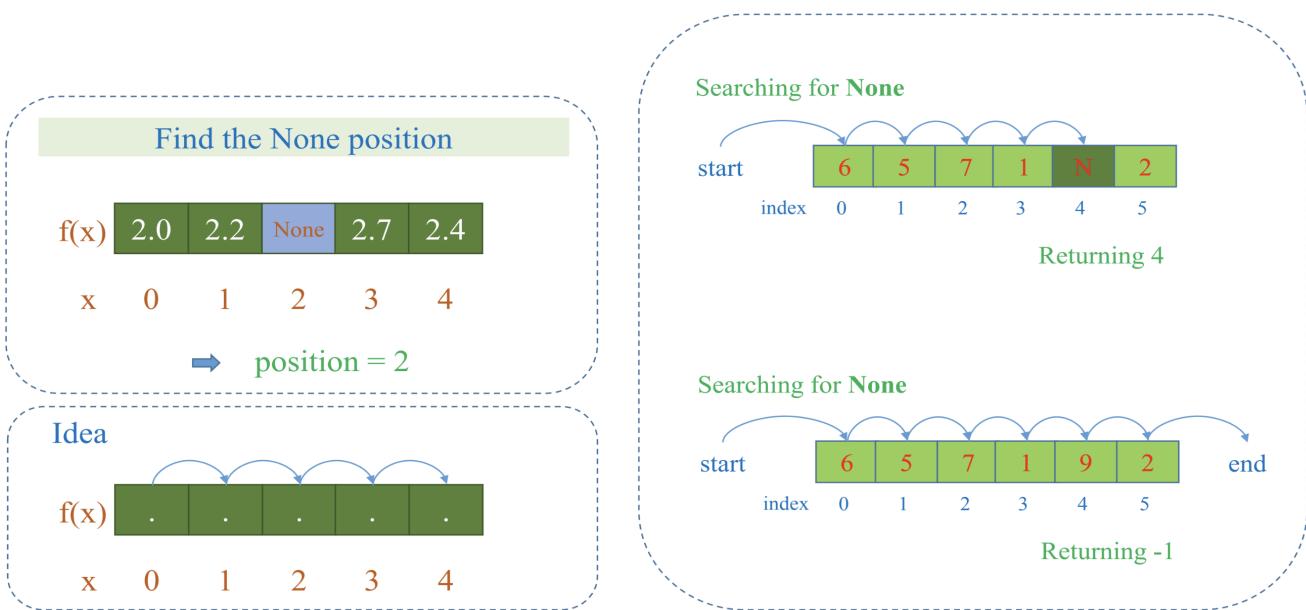
- **Tôi thích AI thích Toán:** [1, 1, 1, 0, 0, 2, 0]
- **Bag-of-Words:** [AI, Toán, Tôi, môn, nhạc, thích, âm]

# Algorithms on List

*Hoàng-Nguyễn Vũ*

## 1. Mô tả:

- **Tìm kiếm (Search)** là thao tác cơ bản thường được sử dụng trong lập trình Python để xác định vị trí hoặc sự tồn tại của một phần tử cần tìm trong list. Ở bài tập này, chúng ta sẽ làm quen với tìm kiếm các giá trị None (Trạng thái không có giá trị. Nó khác với các giá trị 0, False, hoặc "", vì những giá trị này thể hiện một ý nghĩa cụ thể) có trong List:
  - **Bước 1:** Duyệt qua từng phần tử trong list và so sánh với giá trị None.
  - **Bước 2:** Nếu tồn tại giá trị None, thì trả về vị trí hiện tại của giá trị None và kết thúc vòng lặp.



## 2. Bài tập:

- Tạo List có tên là `lst_data = [1, 1.1, None, 1.4, None, 1.5, None, 2.0]. Sau đó, áp dụng phương pháp tìm kiếm để tìm vị trí có giá trị None có trong lst_data theo 2 cách: tìm vị trí None đầu tiên, và tìm tất cả vị trí có giá trị None`

```
1 lst_data = [''' Your Code Here ''']
2 # Your code here
```

### Output:

- Vị trí `None` đầu tiên: 2 - Danh sách vị trí có giá trị `None`: [2, 4, 6]

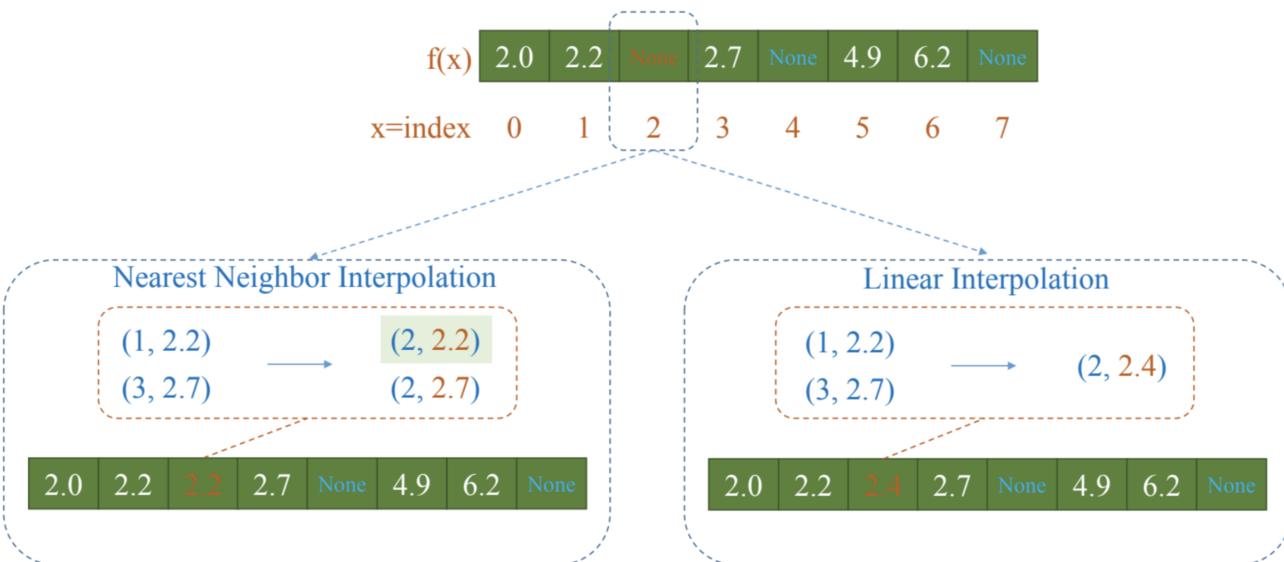
# Interpolation for List (Time-series)

*Hoàng-Nguyễn Vũ*

## 1. Mô tả:

- **Nội suy (Interpolate)** là kỹ thuật dự đoán giá trị tại một điểm chưa biết dựa trên các giá trị đã biết tại các điểm lân cận. Nó được sử dụng trong nhiều lĩnh vực như khoa học, kỹ thuật, kinh tế,... Có nhiều phương pháp nội suy khác nhau, trong đó 2 phương pháp đơn giản và dễ tiếp cận nhất là: **Láng giềng gần nhất (Nearest Neighbor)** và **Nội suy tuyến tính**. Ở bài tập này, chúng ta sẽ tiếp cận với phương pháp Nearest Neighbor (NN). Cách thức hoạt động của phương pháp NN như sau:

- **Bước 1:** Xác định điểm dữ liệu lân cận nhất với điểm cần dự đoán.
- **Bước 2:** Gán giá trị của điểm dữ liệu lân cận nhất cho điểm cần dự đoán.



## 2. Bài tập:

- Tạo List có tên là  $lst\_data = [1, 1.1, \text{None}, 1.4, \text{None}, 1.5, \text{None}, 2.0]$ . Sau đó, áp dụng phương pháp nội suy **Nearest Neighbor** để gán giá trị **None** có trong  $lst\_data$

```
1 lst_data = [''' Your Code Here ''']
2 # Your code here
```

**Output:**

- [1, 1.1, 1.1, 1.4, 1.4, 1.5, 1.5, 2.0]

# 2D List

*Hoàng-Nguyễn Vũ*

## 1. Mô tả:

- **2D List** hay còn gọi là list hai chiều, là một cấu trúc dữ liệu trong Python cho phép lưu trữ dữ liệu dạng bảng. Nó bao gồm các list con, mỗi list con là một hàng trong bảng. Các ứng dụng của 2D List được sử dụng như: Lưu trữ dữ liệu ở dạng bảng, tạo ma trận, biểu diễn đồ thị, xử lý dữ liệu ảnh, ... Để khởi tạo 1 List hai chiều trong python, ta có thể sử dụng đoạn code sau đây:

```
1 list_2d = [
2     [1, 2, 3],
3     [4, 5, 6],
4     [7, 8, 9],
5 ]
```

	Column 0	Column 1	Column 2	Column 3
Row 0	a[ 0 ][ 0 ]	a[ 0 ][ 1 ]	a[ 0 ][ 2 ]	a[ 0 ][ 3 ]
Row 1	a[ 1 ][ 0 ]	a[ 1 ][ 1 ]	a[ 1 ][ 2 ]	a[ 1 ][ 3 ]
Row 2	a[ 2 ][ 0 ]	a[ 2 ][ 1 ]	a[ 2 ][ 2 ]	a[ 2 ][ 3 ]

The diagram illustrates a 2D list structure as a 3x4 grid of cells. The columns are labeled "Column 0", "Column 1", "Column 2", and "Column 3". The rows are labeled "Row 0", "Row 1", and "Row 2". The cells contain the expression "a[ <row> ][ <column> ]" where <row> and <column> are indices ranging from 0 to 3. Three arrows point to specific cells to explain indexing: one arrow points to the first column of Row 1 with the label "Column index"; another arrow points to the first row of Column 1 with the label "Row index"; and a third arrow points to the first cell of the array with the label "Array name".

## 2. Bài tập:

- Tạo List 2D có tên là *lst\_data* có dạng 3 x 3, gồm các số từ 1 đến 9, ứng với các vị trí trong List. Sau đó tạo list khác có tên *lst\_sub\_data* là 2D List để lưu giá trị tại vị trí index thứ 0 và thứ 2 của *lst\_data* (Chỉ sử dụng For). In ra màn hình kết quả của *lst\_sub\_data*

```
1 lst_data = [ ''' Your Code Here ''' ]
2 # Your code here
```

**Output:** [[1, 3], [4, 6], [7, 9]]

# Matrix representation using List

*Hoàng-Nguyễn Vũ*

## 1. Mô tả:

- **Ma trận** là một công cụ toán học hữu ích để biểu diễn và thao tác với dữ liệu. Nó được cấu tạo bởi các hàng và cột, chứa các giá trị số được sắp xếp theo thứ tự. Ma trận có thể được sử dụng để biểu diễn nhiều loại dữ liệu khác nhau, chẳng hạn như: dữ liệu hình ảnh, dữ liệu ngôn ngữ, v.v... Dưới đây là một số phép toán cơ bản trong ma trận:
  - **Cộng/Trừ ma trận:** Hai ma trận có cùng kích thước có thể được cộng/trừ với nhau để tạo ra một ma trận mới
  - **Tích vô hướng ma trận:** Dot product của hai ma trận A và B có kích thước tương thích ( $m \times n$  và  $n \times p$ ) là ma trận C có kích thước  $m \times p$

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \text{ và } B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

$$A \pm B = \begin{bmatrix} a_{11} \pm b_{11} & a_{12} \pm b_{12} \\ a_{21} \pm b_{21} & a_{22} \pm b_{22} \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} \times \begin{bmatrix} w \\ z \end{bmatrix} = \begin{bmatrix} a.w + b.z \\ c.w + d.z \\ e.w + f.z \end{bmatrix}$$

2. **Bài tập:** Cho 2 ma trận sau:  $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$  và  $B = \begin{bmatrix} 2 & 4 & 6 \\ 1 & 3 & 5 \\ 1 & 0 & 1 \end{bmatrix}$ . Sử dụng Python, không dùng thư viện numpy

**Câu 1.** Hãy tính tổng và hiệu 2 ma trận  $A + B$  và  $A - B$

**Câu 2.** Hãy tính dot product 2 ma trận A và B

```
1 mat_a = [ ''' Your Code Here ''' ]
2 mat_b = [ ''' Your Code Here ''' ]
3 # Your code here
```

### Output:

- **Tổng:**  $[[3, 6, 9], [5, 8, 11], [8, 8, 10]]$
- **Hiệu:**  $[[1, -2, -3], [3, 2, 1], [6, 8, 8]]$
- **Dot Product:**  $[[7, 10, 19], [19, 31, 55], [31, 52, 91]]$

# List Comprehension

*Hoàng-Nguyễn Vũ*

## 1. Mô tả:

- **List comprehension** là một cú pháp ngắn gọn và mạnh mẽ trong Python để tạo ra một danh sách mới từ một danh sách hoặc tập hợp dữ liệu có sẵn. Cú pháp này giúp bạn tiết kiệm thời gian và viết code ngắn gọn hơn so với việc sử dụng vòng lặp for truyền thống.

### - Ưu điểm:

- + **Giảm thiểu số lượng code:** giúp code ngắn gọn và dễ đọc hơn so với sử dụng vòng lặp for truyền thống.

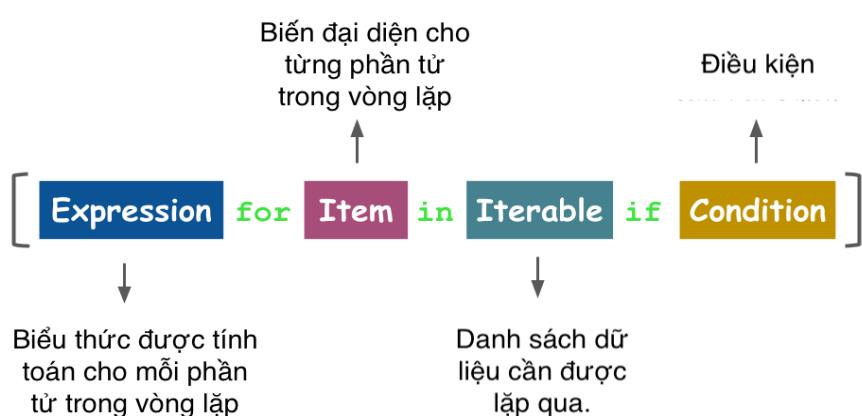
- + **Dễ sử dụng:** có cú pháp đơn giản và dễ học.

### - Nhược điểm:

- + **Khó đọc:** có thể khó đọc và khó hiểu hơn so với vòng lặp for truyền thống trong một số trường hợp.

- + **Hạn chế về chức năng:** không thể thực hiện một số thao tác mà vòng lặp for truyền thống có thể làm được

Cấu trúc của List comprehension như sau:



2. **Bài tập:** Trong NLP, chúng ta cần loại bỏ 1 số từ không quan trọng (stopwords) ra khỏi câu để tránh gây nhiễu trong việc xử lý. Hãy loại bỏ các từ có trong `stop_words = ["I", "love", "and", "to"]` câu đầu vào "`I love AI and listen to music`". Hãy áp dụng **List comprehension** và For truyền thống để thực hiện

```

1 stop_words = ["I", "love", "and", "to"]
2 input = "I love AI and listen to music"
3 # Your code here
    
```

**Output:** ['AI', 'listen', 'music']

# Mathematic with List

*Hoang-Nguyen Vu*

## 1 Chuẩn hóa dữ liệu (Normalization)

**Mô tả:** Trong Machine Learning, dữ liệu có thể có phạm vi giá trị rất lớn, làm cho mô hình khó học được quy luật chung. Vì vậy, cần chuẩn hóa dữ liệu về khoảng [0, 1] theo công thức:

$$X' = \frac{X - \min(X)}{\max(X) - \min(X)}$$

Điều này giúp dữ liệu đồng nhất, tránh hiện tượng một số giá trị quá lớn lấn át các giá trị nhỏ hơn trong mô hình học máy.

```

1 def normalize(lst_data):
2     ## Your code here
3
4 test_cases = [
5     [11, 18, 24, 30, 36],
6     [50, 100, 150, 200, 250],
7     [3, 5, 7, 9, 11]
8 ]
9
10 for i, test in enumerate(test_cases):
11     result = normalize(test)
12     print(f"Test {i+1}: {result}")

```

**Output:**

Test 1: [0.0, 0.28, 0.52, 0.76, 1.0]  
 Test 2: [0.0, 0.25, 0.5, 0.75, 1.0]  
 Test 3: [0.0, 0.25, 0.5, 0.75, 1.0]

## 2 Trung bình động (Moving Average)

**Mô tả:** Trung bình động làm mượt dữ liệu, giúp phát hiện xu hướng rõ ràng hơn. Công thức:

$$MA_t = \frac{1}{k} \sum_{i=t-k+1}^t X_i$$

```

1 def moving_average(lst, k):
2     # Your code here
3
4 test_cases = [
5     ([1, 2, 3, 4, 5, 6, 7, 8, 9], 3),
6     ([10, 20, 30, 40, 50, 60, 70], 4),
7     ([5, 10, 15, 20, 25], 2)

```

```

8 ]
9
10 for i, (test, k) in enumerate(test_cases):
11     result = moving_average(test, k)
12     print(f"Test {i+1}: {result}")

```

**Output:**

Test 1: [2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0]

Test 2: [25.0, 35.0, 45.0, 55.0]

Test 3: [7.5, 12.5, 17.5, 22.5]

### 3 Tích vô hướng của hai vector

**Mô tả:** Tích vô hướng dùng để đo độ tương đồng giữa hai vector:

$$v_1 \cdot v_2 = \sum_{i=1}^n v_1[i] \times v_2[i]$$

```

1 def dot_product(v1, v2):
2     ## Your code here ##
3
4 test_cases = [
5     ([1, 2, 3], [4, 5, 6]),
6     ([2, 4, 6], [1, 3, 5]),
7     ([0, 1, 2], [3, 4, 5])
8 ]
9
10 for i, (v1, v2) in enumerate(test_cases):
11     result = dot_product(v1, v2)
12     print(f"Test {i+1}: {result}")

```

**Output:**

Test 1: 32

Test 2: 44

Test 3: 14

### 4 Mô phỏng Perceptron

**Mô tả:** Perceptron thực hiện tổng trọng số và áp dụng hàm kích hoạt ReLU:

$$y = \sum w_i \cdot x_i + bias$$

$$\text{Output} = \text{ReLU}(y)$$

```

1 def perceptron_relu(weights, inputs, bias):
2     ## Your code here ##
3
4 test_cases = [

```

```
5 ([0.5, -0.6, 0.8], [1.0, 0.0, 1.0], -0.3),  
6 ([0.2, 0.5, -0.4], [1.0, 2.0, -1.0], 0.1),  
7 ([1.0, -1.0, 0.5], [0.5, 1.0, -0.5], -0.2)  
]  
9  
10 for i, (weights, inputs, bias) in enumerate(test_cases):  
11     result = perceptron_relu(weights, inputs, bias)  
12     print(f"Test {i+1}: {result}")
```

**Output:**

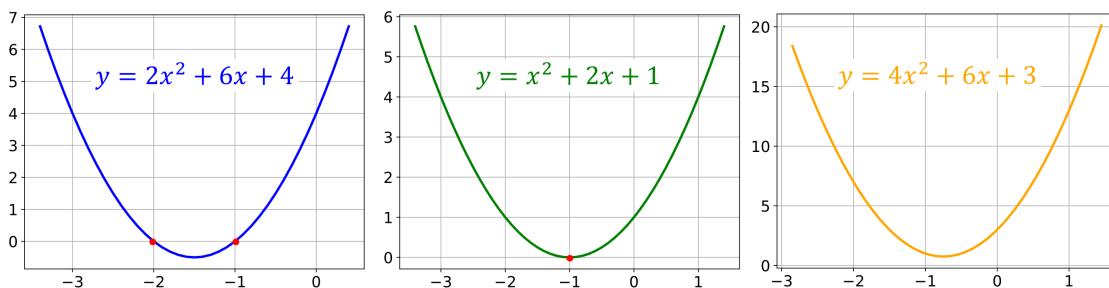
Test 1: 1.0  
Test 2: 1.7000000000000002  
Test 3: 0.0

## Basic Python - If-Else

*Trung-Trực Trần và Hoàng-Nguyễn Vũ*

### 1. Bài tập:

- Cho một phương trình bậc 2 có dạng  $y = ax^2 + bx + c = 0$  (a khác 0). Hãy hoàn thành function giải phương trình bậc 2 trên, biết  $\Delta = b^2 - 4ac$ .



### 2. Ý tưởng: Tính nghiệm phương trình bậc 2 - làm quen với câu lệnh If/Else

- Để tính được nghiệm, chúng ta dựa vào công thức sau đây:
  - Nếu  $\Delta$  lớn hơn 0, phương trình có 2 nghiệm phân biệt là  $x_1 = \frac{-b-\sqrt{\Delta}}{2a}$  và  $x_2 = \frac{-b+\sqrt{\Delta}}{2a}$
  - Nếu  $\Delta$  bằng 0 thì phương trình có nghiệm kép  $x_1 = x_2 = \frac{-b}{2a}$
  - Nếu  $\Delta$  bé hơn 0 thì phương trình vô nghiệm

```

1 def quadratic_equation(a, b, c):
2     x1=0
3     x2=0
4
5     # Your code here #####
```

#### Ví dụ:

- test case 1:** với `quadratic_equation(a=2, b=6, c=4)`, kết quả gồm  $x_1 = -1$  và  $x_2 = -2$ .
- test case 2:** với `quadratic_equation(a=1, b=2, c=1)`, kết quả gồm  $x_1 = x_2 = -1$
- test case 3:** với `quadratic_equation(a=4, b=6, c=3)`, kết quả là phương trình vô nghiệm.

**Mở rộng:** Trường hợp  $a = 0$ , hàm số sẽ thành phương trình bậc 1 có dạng:  $y = bx + c = 0$ , trường hợp này các bạn có thể mở rộng để bài toán tổng quát hơn.

# Basic Python - For-Loop

*Hoàng-Nguyễn Vũ*

## 1 Cấu trúc vòng lặp for-loop trong Python

Vòng lặp **for-loop** trong Python được sử dụng để lặp qua các phần tử trong một chuỗi (sequence) như danh sách, chuỗi ký tự, tuple, từ điển, hoặc dải giá trị số (**range**).

### Cấu trúc chung

Cú pháp cơ bản của **for-loop** trong Python:

```
1 for variable in sequence:  
2     # Thực hiện các lệnh bên trong vòng lặp
```

### Ví dụ 1: Lặp qua một danh sách

```
1 numbers = [1, 2, 3, 4, 5]  
2 for num in numbers:  
3     print(num)
```

### Ví dụ 2: Lặp qua một dải số

```
1 for i in range(5):    # range(5) tạo ra các số từ 0 đến 4  
2     print(i)
```

### Ví dụ 3: Lặp qua một chuỗi ký tự

```
1 for char in "Python":  
2     print(char)
```

## 2 Bài tập

### 2.1 Tính lãi suất tiền gửi ngân hàng - Dùng vòng lặp for

- + Số e (còn gọi là hằng số Euler) là một số vô tỷ xuất hiện trong nhiều lĩnh vực toán học và khoa học, bao gồm cả lĩnh vực tài chính. Trong ngân hàng, số e được sử dụng để tính lãi suất kép, một phương pháp tính lãi suất trong đó tiền lãi được cộng vào số tiền gốc để tính lãi cho các kỳ hạn tiếp theo. Công thức tổng quát của số e như sau:  $\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n$



- + Giả sử bạn có 1 Dollar, và bạn gửi vào ngân hàng và được nhận lãi mỗi ngày, vậy điều sẽ xảy ra nếu bạn gửi vào ngân hàng trong 1 năm thì tiền bạn nhận được là bao nhiêu ? Để trả lời câu hỏi này, chúng ta làm như sau:

- Bước 1: Tiền nhận được sau 1 ngày gửi:  $1 + \frac{1}{365}$
- Bước 2: lặp lại cách tính này cho 1 năm, ta sẽ được công thức của e:  $\left(1 + \frac{1}{n}\right)^n$

```
1 def compute_interest(money, period):
2     ##### Your code here #####
```

#### Ví dụ:

- Test case 1: `compute_interest(1, 12) → 2.613`
- Test case 2: `compute_interest(1, 365) → 2.714`
- Test case 3: `compute_interest(1, 730) → 2.716`

## 2.2 Thuật toán sàng số nguyên tố Eratosthenes

**Mô tả:** Sử dụng thuật toán sàng Eratosthenes để kiểm tra xem một số nguyên  $n$  (được nhập từ người dùng) có phải là số nguyên tố hay không.

**Algorithm 1** Kiểm tra số nguyên tố sử dụng sàng Eratosthenes

**Require:** Một số nguyên  $n \geq 0$

**Ensure:** Trả về True nếu  $n$  là số nguyên tố, ngược lại trả về False

```

1: if  $n < 2$  then
2:   return False                                ▷ Số nhỏ hơn 2 không phải là số nguyên tố
3: end if
4: Tạo một danh sách prime với  $n + 1$  phần tử, tất cả được gán True
5: Gán prime[0] và prime[1] là False            ▷ 0 và 1 không phải số nguyên tố
6: for  $i \leftarrow 2$  to  $\sqrt{n}$  do
7:   if prime[i] = True then
8:     for  $j \leftarrow i^2$  to  $n$  step  $i$  do
9:       prime[j] ← False                         ▷ Dánh dấu tất cả các bội số của  $i$  từ  $i^2$  đến  $n$ 
10:      end for
11:    end if
12:  end for
13: end for
14: return prime[n]                                ▷ Kiểm tra giá trị của prime[n]

```

```

1 def is_prime_eratosthenes(n):
2   ##### Your code here #####

```

Ví dụ:

- Test case 1: is\_prime\_eratosthenes(7) → True
- Test case 2: is\_prime\_eratosthenes(10) → False
- Test case 3: is\_prime\_eratosthenes(2) → True
- Test case 4: is\_prime\_eratosthenes(1) → False

# Basic Python - For-Loop

*Trung-Trực Trần và Hoàng-Nguyễn Vũ*

## 1. Bài tập:

- Nhập vào số nguyên dương n, chạy vòng lặp từ 1 đến n và in ra màn hình theo yêu cầu của đề bài.

## 2. Ý tưởng: Sử dụng vòng lặp For và câu lệnh If-else

- Điều kiện yêu cầu phụ thuộc vào biến chạy của vòng lặp:
  - Nếu i chia hết cho 3 thì in ra màn hình chữ cái Fizz.
  - Nếu i chia hết cho 5 thì in ra màn hình chữ cái Buzz.
  - Nếu i chia hết cho 5 và 3 thì in ra màn hình từ Fizz-Buzz.
  - Các phần tử i còn lại thì in ra chính nó.

```
1 def fizz_buzz(n):
2     for i in range(1, n + 1):
3         ##### Your code here #####
```

### Ví dụ:

- **test case 1:** input: n=3 output: 1, 2, Fizz.
- **test case 2:** input: n=5 output: 1, 2, Fizz, 4, Buzz.
- **test case 3:** input: n=15 output: 1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, Fizz-Buzz.

# Basic Python - For-Loop

*Trung-Trực Trần*

## 1. Bài tập:

- Nhập vào số nguyên n, tính giai thừa của số nguyên đó rồi in ra màn hình nếu số.

## 2. Ý tưởng: Sử dụng vòng lặp For

- Tạo 2 biến để lưu kết quả của giai thừa:
  - Biết giai thừa  $n! = n * (n-1) * (n-2) * \dots * 1$ .

```
1 def factorial(n):
2     gai_thua=1
3     for i in range(1, n + 1):
4         # Your code here
```

### Ví dụ:

- **test case 1:** input: n=5 output: 120.
- **test case 2:** input: n=4 output: 24.
- **test case 3:** input: n=3 output: 6.

# Basic Python - For-Loop

*Trung-Trực Trần và Hoàng-Nguyễn Vũ*

## 1. Bài tập:

- Nhập vào số nguyên dương n, chạy vòng lặp từ 1 đến n in ra màn hình tổng của các số chẵn và tổng của các số lẻ.

## 2. Ý tưởng: Sử dụng vòng lặp For và câu lệnh If-else

- Tạo 2 biến tổng cho Chẵn và Lẻ:
  - Nếu i là số chẵn thì cộng i vào biến chẵn.
  - Nếu i là số lẻ thì cộng i vào biến lẻ.

```
1 def calculate_total(n):
2     odd = 0
3     even = 0
4     for i in range(1, n + 1):
5         # Your code here
```

Ví dụ:

- **test case 1:** input: n=3 output: odd=2, even=4.
- **test case 2:** input: n=5 output: odd=6, even=9.
- **test case 3:** input: n=2 output: odd=2, even=1.

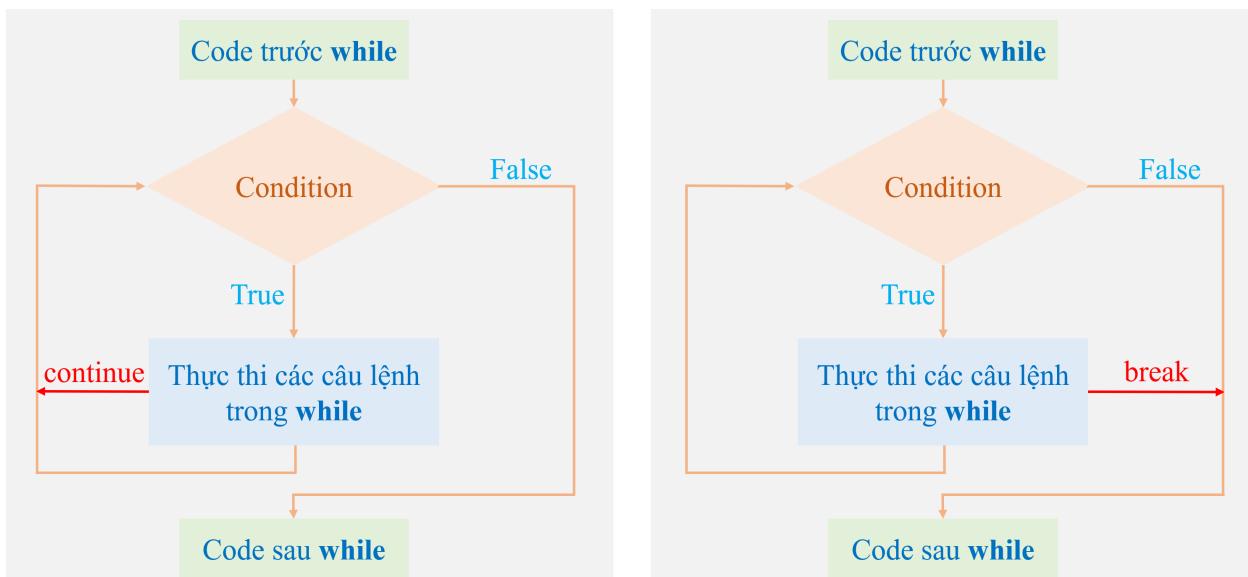
# Basic Python

## While Loop + continue and break keywords

*Hoang-Nguyen Vu*

### 1. Mô tả:

- Trong Python, câu lệnh **while** được sử dụng để tạo một vòng lặp, trong đó các biểu thức được kiểm tra và nếu điều kiện đúng, khối mã lệnh bên trong vòng lặp sẽ được thực thi. Các lệnh **break** và **continue** được sử dụng để kiểm soát luồng của vòng lặp.



Hình 1: Vòng lặp while kết hợp với từ khóa continue và break.

- Câu lệnh **break** được sử dụng để thoát khỏi vòng lặp ngay lập tức.
  - Câu lệnh **continue** được sử dụng để bỏ qua phần còn lại của vòng lặp và chuyển đến lần lặp tiếp theo.
- 2. Bài tập:** Cài đặt hàm `find_divisible_number(a)` tìm số nguyên dương nhỏ nhất lớn hơn 100 và chia hết cho số nguyên dương `a`

```

1 def find_divisible_number(a):
2     """
3         Find the smallest integer that is greater than 100 and divisible
4         by the integer a
5     """
6
7     #TODO: Your code here
  
```

### Ví dụ:

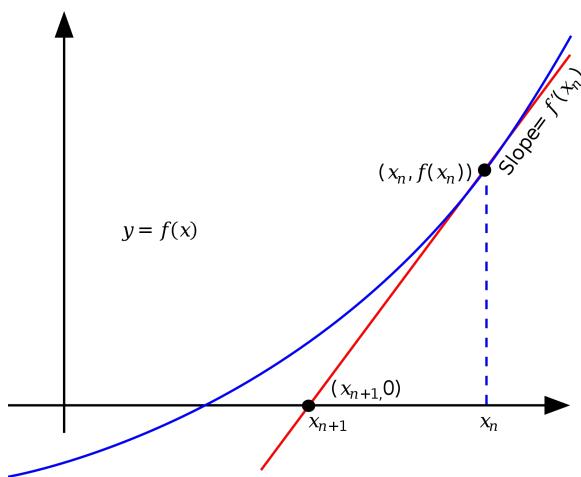
- Test case 1: `find_divisible_number(5) → 105`
- Test case 2: `find_divisible_number(17) → 102`

# Basic Python - While-Loop Exercise

*Hoàng-Nguyễn Vũ*

## 1. Mô tả:

- Phương pháp Newton (Newton's Method), còn được gọi là phương pháp Newton-Raphson, là một phương pháp số học để tìm gần đúng của các nghiệm của một hàm số thực. Cụ thể, nó thường được sử dụng để tìm gần đúng của các nghiệm của phương trình  $f(x) = 0$ .
- Ngoài ứng dụng trong tìm nghiệm của một hàm số, phương pháp Newton còn có ứng dụng trong máy học (Machine learning) trong việc tìm nghiệm của đạo hàm của hàm loss. Tuy nhiên đây là phương pháp không phổ biến bằng thuật toán gradient descent.



- Ở bài này, chúng ta sẽ dùng phương pháp Newton để tính căn bậc hai cho một số dương  $a$ . Chúng ta thực hiện các bước sau:
  - Khởi tạo giá trị  $x_0 = a$ ,  $n = 0$  và cho trước giá trị  $\varepsilon$  (thực ra  $x_0$  có thể nhận bất kỳ giá trị dương nào). Tiếp đó, ta sẽ đi xây dựng hàm  $f(x) = x^2 - a$ . Ở đây, ta xem  $x_n$  (ở bước hiện tại đang là  $x_0$ ) chính là lời giải cho bài toán tính căn bậc hai của  $a$ .
  - Cải thiện xấp xỉ  $x_n$  bằng xấp xỉ  $x_{n+1}$  theo công thức tổng quát như sau:  

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$
  - So sánh xấp xỉ  $x_{n+1}$  và  $x_n$ . Nếu  $|x_{n+1} - x_n| < \varepsilon$ , ta sẽ dừng việc cải thiện xấp xỉ, và trả về kết quả căn bậc hai của  $a$  là  $x_{n+1}$ . Ngược lại thực hiện tiếp bước (b) với  $n = n + 1$ .

2. **Bài tập:** Cài đặt hàm find\_squared\_root(a) tìm căn bậc hai cho một số  $a$  bất kì với  $\varepsilon = 0.001$ .

```
1 def find_squared_root(a):
2     """Find the squared root of number a"""
3     EPSILON = 0.001
4
5     #TODO: Your code here
6
```

**Ví dụ:**

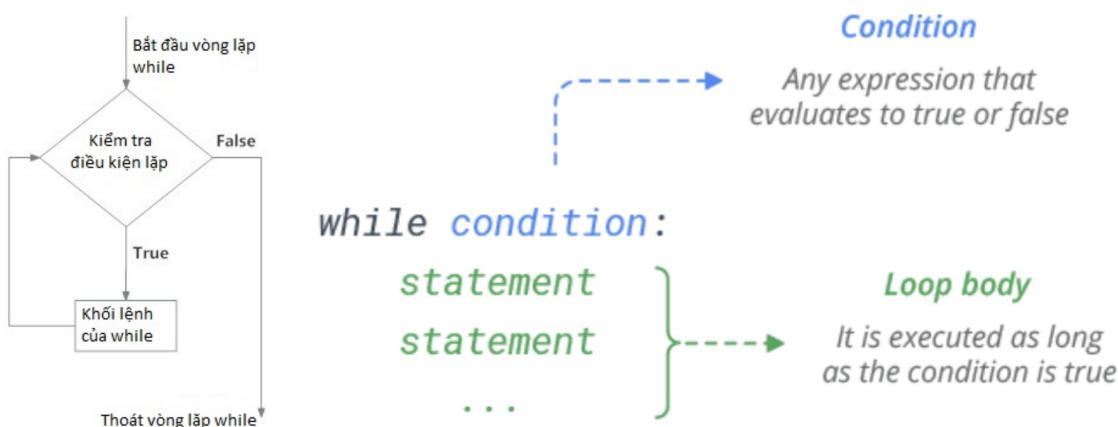
- Test case 1:  $\text{find_squared_root}(2) \rightarrow 1.4142135623746899$
- Test case 2:  $\text{find_squared_root}(3) \rightarrow 1.7320508100147276$

# Basic Python - While-Loop

*Hoàng-Nguyễn Vũ*

## 1. Mô tả:

- Sự ngẫu nhiên là một đặc điểm cơ bản của các hiện tượng trong thực tế. Nó thể hiện qua việc kết quả của một thí nghiệm không thể dự đoán trước được một cách chính xác. **Ví dụ:** Chọn ngẫu nhiên hai số a và b sao cho tổng của a + b bằng 40, câu hỏi đặt ra rằng, chúng ta phải tạo ngẫu nhiên bao nhiêu lần để thỏa mãn điều kiện trên ?
- Cấu trúc vòng lặp While-Loop:** Vòng lặp while là một cấu trúc điều khiển trong Python cho phép thực thi một khối mã nhiều lần miễn là điều kiện cho trước vẫn còn đúng. Cấu trúc câu lệnh như sau:



## 2. Bài tập: về sự ngẫu nhiên - Dùng vòng lặp While

- Chọn ngẫu nhiên hai số a và b thuộc từ 1 đến 20 sao cho tổng của a + b bằng 40, câu hỏi đặt ra rằng, chúng ta phải tạo ngẫu nhiên bao nhiêu lần để thỏa mãn điều kiện trên ?

```

1 import random
2 def random_number_with_condition(total):
3     # Set a random value that is the same between devices
4     random.seed(0)
5     # Your code here
6

```

### Ví dụ:

- Test case 1: `random_number_with_condition(40)` → 266 lần
- Test case 2: `random_number_with_condition(20)` → 32 lần
- Test case 3: `random_number_with_condition(35)` → 96 lần

# Basic Python - File

*Hoàng-Nguyễn Vũ*

## 1. Mô tả:

- **Đọc file** là thao tác lấy dữ liệu từ file vào chương trình Python để xử lý. **Ghi file** là thao tác lưu dữ liệu từ chương trình Python vào file. Có nhiều cách để đọc và ghi file trong Python:

- **Cách 1: Sử dụng hàm open()**: Hàm open() được sử dụng để mở file. Sau khi mở file, bạn có thể sử dụng các phương thức khác để đọc hoặc ghi dữ liệu vào file, cuối cùng ta cần phải đóng file lại sau khi thực thi xong các quá trình trên. Ví dụ:

```

1 # Read File
2 f = open("test.txt", "r")
3 data = f.read()
4 f.close()
5
6 print(data)
7
8 # Write File
9 f = open("test.txt", "w")
10 f.write("I Love AI Vietnam")
11 f.close()
```

- **Cách 2: Sử dụng câu lệnh with**: Câu lệnh with giúp bạn đảm bảo rằng file được đóng đúng cách sau khi sử dụng. Ví dụ:

```

1 # Read File
2 with open("test.txt", "r") as f:
3     data = f.read()
4 print(data)
5
6 # Write File
7 with open("test.txt", "w") as f:
8     f.write("I Love AI Vietnam")
```

## 2. Bài tập:

- **Câu 1:** Tạo List có tên *lst\_data* gồm các số từ 1 đến 10, sau đó ghi toàn bộ list trên vào file có tên: *data.txt* với nội dung là 1 chuỗi số từ list trên nối với nhau bằng dấu -
- **Câu 2:** Đọc file *data.txt* vừa tạo ở câu 1 và lưu vào List mới có tên *lst\_filter* gồm các số chia hết cho 3

### Output:

- **Câu 1:** 1-2-3-4-5-6-7-8-9-10
- **Câu 2:** [3,6,9]

# Basic Python - File

*Hoàng-Nguyễn Vũ*

## 1. Mô tả:

- Ở bài tập trước, chúng ta đã làm quen với thao tác đọc ghi file, bài tập này sẽ giúp chúng ta hiểu rõ và áp dụng cơ bản trong việc phân tích dữ liệu có trong file:



## 2. Bài tập: Dữ liệu dataset

- **Câu 1:** Hãy đọc file *data.txt* có trong dataset phía trên và lưu vào biến *data* sau khi loại bỏ các ký tự \n và thay thế bằng khoảng trắng, đồng thời chuyển về chữ cái thường toàn bộ văn bản
- **Câu 2:** Phân tích văn bản trên và lưu vào biến có tên *distinct\_words* các chữ cái duy nhất trong câu
- **Câu 3:** Đếm số lần từng chữ trong *distinct\_words* xuất hiện trong văn bản. Và cho biết chữ nào xuất hiện **nhiều nhất** và **ít nhất**.

```

1 # Read File
2 with open("data.txt", "r") as f:
3     # Your Code Here

```

### Output:

- **Câu 1:** he who conquers himself.....people get what they want
- **Câu 2:** 'a', 'again', 'and', ..., 'with', 'you', 'your'
- **Câu 3:**
  - + "thought" in data is 1
  - + "you" in data is 4
  - + "a" is most frequent word
  - + "makes" is the least common word

# Basic Python - File

*Hoàng-Nguyễn Vũ*

## 1. Mô tả:

**Pandas** là một thư viện mã nguồn mở được viết bằng Python, chuyên dụng cho việc phân tích dữ liệu và thao tác dữ liệu. Nó cung cấp một loạt các công cụ mạnh mẽ để giúp bạn:

- Đọc và ghi dữ liệu từ nhiều nguồn khác nhau, bao gồm tệp CSV, Excel, cơ sở dữ liệu SQL, v.v.
- Làm sạch và chuẩn bị dữ liệu cho phân tích.
- Thực hiện các phép toán thống kê trên dữ liệu.
- Tạo biểu đồ và hình ảnh để trực quan hóa dữ liệu.

## 2. Bài tập: Dữ liệu dataset McDonald

- **Phân tích dữ liệu thực đơn của McDonald** Hãy đọc file *menu.csv* có trong dataset phía trên và lưu vào biến *data* với thư viện Pandas. Sau đó sử dụng hàm *describe()*, *info()*, *head()*, *tail()* để thống kê tập dataset. **Lưu ý:** Lấy 5 phần tử đầu tiên và 10 phần tử cuối cùng khi dùng hàm *head* và *tail*

	Calories	Calories from Fat	Total Fat	Total Fat (% Daily Value)	Saturated Fat	Saturated Fat (% Daily Value)
<b>count</b>	260.000000	260.000000	260.000000	260.000000	260.000000	260.000000
<b>mean</b>	368.269231	127.096154	14.165385	21.815385	6.007692	29.965385
<b>std</b>	240.269886	127.875914	14.205998	21.885199	5.321873	26.639209
<b>min</b>	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
<b>25%</b>	210.000000	20.000000	2.375000	3.750000	1.000000	4.750000
<b>50%</b>	340.000000	100.000000	11.000000	17.000000	5.000000	24.000000
<b>75%</b>	500.000000	200.000000	22.250000	35.000000	10.000000	48.000000
<b>max</b>	1880.000000	1060.000000	118.000000	182.000000	20.000000	102.000000

Hình 1: Kết quả của hàm *describe()*

1 # Info		
<class 'pandas.core.frame.DataFrame'>		
RangeIndex: 260 entries, 0 to 259		
Data columns (total 24 columns):		
# Column	Non-Null Count	Dtype
0 Category	260 non-null	object
1 Item	260 non-null	object
2 Serving Size	260 non-null	object
3 Calories	260 non-null	int64
4 Calories from Fat	260 non-null	int64
5 Total Fat	260 non-null	float64
6 Total Fat (% Daily Value)	260 non-null	int64

Hình 2: Kết quả của hàm *info()*

# Basic Python - Work with Text Data

*Hoàng-Nguyễn Vũ*

## 1. Mô tả: Làm quen với thư viện gensim



- **Gensim** là một thư viện Python mã nguồn mở được sử dụng để xử lý ngôn ngữ tự nhiên (NLP) và học máy. Nó cung cấp một bộ công cụ mạnh mẽ để xử lý các tác vụ NLP phổ biến, bao gồm:

Bảng 1: Tính năng, ưu điểm và ứng dụng của thư viện Gensim

<b>Tính năng</b>	
Phân tích văn bản	Sử dụng để chia nhỏ văn bản thành các thành phần cấu tạo của nó, chẳng hạn như từ, cụm từ và câu.
Mô hình hóa ngôn ngữ	Sử dụng để tạo các mô hình ngôn ngữ có thể dự đoán các từ tiếp theo trong một chuỗi hoặc xác định mối quan hệ giữa các từ.
Tìm kiếm thông tin	Sử dụng để xây dựng các hệ thống tìm kiếm thông tin có thể truy xuất các tài liệu liên quan đến một truy vấn nhất định.
Phân loại văn bản	Phân loại văn bản vào các chủ đề hoặc thẻ loại khác nhau.
<b>Ưu điểm</b>	
Hiệu quả	Được tối ưu hóa cho hiệu suất cao và có thể xử lý lượng dữ liệu lớn một cách hiệu quả.
Dễ sử dụng	Cung cấp API đơn giản và dễ hiểu.
Linh hoạt	Được sử dụng để xây dựng nhiều loại hệ thống NLP khác nhau.
<b>Ứng dụng</b>	
Xử lý ngôn ngữ tự nhiên	Sử dụng để thực hiện các tác vụ NLP phổ biến như phân tích văn bản, mô hình hóa ngôn ngữ, tìm kiếm thông tin, chủ đề và phân loại văn bản.
Học máy	Sử dụng để xây dựng các hệ thống học máy cho các tác vụ NLP như tóm tắt văn bản, dịch máy và chatbot.

- **Cách cài đặt và sử dụng một số tính năng:**

- Để cài đặt thư viện gensim, ta sẽ cài thông qua câu lệnh:

```
1 !pip install gensim
```

- Cách sử dụng các tính năng chính của thư viện:

- + **Phân tích văn bản:**

```
1 from gensim import corpora
2 from nltk.tokenize import word_tokenize
3 # Tạo một danh sách các câu
4 sentences = [
5     "Tôi thích AI",
6     "Tôi thích âm nhạc"
7 ]
8
9 # Chia nhỏ các câu thành các từ
10 words = [word_tokenize(sentence) for sentence in sentences]
11
12 # Tạo một từ điển
13 dictionary = corpora.Dictionary(words)
14
15 # Chuyển đổi các câu thành các vector bag-of-words
16 bow_corpus = [dictionary.doc2bow(sentence) for sentence in
17               words]
18 print(f'Corpus : {bow_corpus}')
19 # In ra các vector bag-of-words
20 for index, vector in enumerate(bow_corpus, 1):
21     print(f'Câu {index} - Vector: {vector}')
```

- + **Kết quả:**

Corpus : [[(0, 1), (1, 1), (2, 1)], [(1, 1), (2, 1), (3, 1), (4, 1)]]  
Câu 1 - Vector: [(0, 1), (1, 1), (2, 1)]  
Câu 2 - Vector: [(1, 1), (2, 1), (3, 1), (4, 1)]

- + **Mô hình hóa ngôn ngữ: Word Similarity**

```
1 from gensim.models import Word2Vec
2
3 # Tạo mô hình Word2Vec
4 model = Word2Vec(words, min_count=1)
5
6 # In ra các từ tương tự với từ "AI"
7 similar_words = model.wv.most_similar("AI")
8 print(f'Từ tương tự với "AI": {similar_words}')
```

+ **Kết quả:** Từ tương tự với "AI": [('nhạc', 0.17018885910511017), ('Tôi', 0.004503022879362106), ('thích', -0.027750348672270775), ('âm', -0.04461711645126343)]

## 2. Bài tập: Làm quen với thư viện Gensim

- **Câu 1:** Thực hiện tạo Bag of Words cho câu sau: ["Robot của kỳ lân công nghệ Figure AI", "Figure AI là startup mới thành lập năm 2022 tại Mỹ nhưng đã tạo ấn tượng mạnh với giới đầu tư"]
- **Câu 2:** In ra các từ tương tự với từ "Robot"

Kết quả:

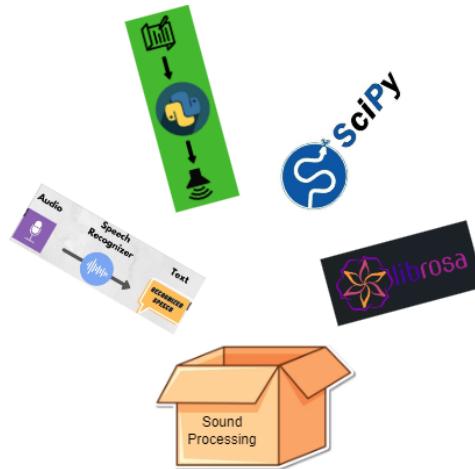
+ **Câu 1: Corpus :** `[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), ..., (24, 1), (25, 1), (26, 1)]`

+ **Câu 2:** Từ tương tự với "Robot": `[('của', 0.2008291482925415), ('lân', 0.17751173675060272), ...]`

# Thư viện xử lý âm thanh

*Hoang-Nguyen Vu*

1. **Giới thiệu:** Trong Python, có một số thư viện phổ biến được sử dụng để xử lý âm thanh, các thư viện này giúp người dùng dễ dàng sử dụng để thao tác với file âm thanh chẳng hạn như đọc, ghi, cắt, nối và xử lý các file, chúng ta cùng tìm hiểu một số thư viện phổ biến hiện nay của Python trong dữ liệu này:



**Librosa:** Thư viện mạnh mẽ cho phân tích và xử lý âm thanh và âm nhạc.

- **librosa.load:** Hàm để đọc file âm thanh.
- **librosa.example:** Hàm để tải một ví dụ âm thanh từ bộ dữ liệu mẫu của Librosa.

**Scipy:** là một thư viện Python mạnh mẽ cho tính toán khoa học và kỹ thuật.

- **scipy.fft:** Hàm để thực hiện biến đổi Fourier.
- **scipy.signal.spectrogram:** Hàm để tạo biểu đồ phổ của tín hiệu.

**SpeechRecognition:** Thư viện để trích xuất thông tin từ dữ liệu âm thanh.

- **recognizer.record:** ghi âm từ nguồn âm thanh đã chọn và trả về dữ liệu âm thanh dưới dạng một đối tượng.
- **recognizer.recognize\_google:** Phương thức này được sử dụng để nhận dạng văn bản từ dữ liệu âm thanh bằng cách sử dụng dịch vụ nhận dạng giọng nói của Google (Cần kết nối internet).

**gtts:** là một thư viện Python cho phép bạn tạo và phát lại giọng nói từ văn bản sử dụng dịch vụ Text-to-Speech của Google.

**Cần các thuộc tính sau để sử dụng thư viện này:**

- lang = 'vi' (Ngôn ngữ trả về).
- output\_filename = 'record.mp3' (Loại file âm thanh đầu ra).
- content = "xin chào mọi người" (đoạn văn bản muốn chuyển thành voice).

## 2. Ví dụ cách sử dụng thư viện xử lý âm thanh.

### Librosa

- Cài đặt thư viện:

```
1 !pip install librosa
```

- Sử dụng thư viện librosa để visualize âm thanh:

```
1 import librosa
2
3 filename = librosa.example('nutcracker')
4 # Load the audio as a waveform `y`
5 # Store the sampling rate as `sr`
6 audio, sr = librosa.load(filename)
7
8 import numpy as np
9 import matplotlib.pyplot as plt
10 time = np.linspace(0, audio.shape[0] / sr, num=audio.shape[0])
11 plt.plot(time, audio, color="blue")
12 plt.xlabel("Time (s)")
13 plt.ylabel("Amplitude (quantized)")
14 plt.title("Wav file visualization")
15 plt.axhline(y=0, color='r', linestyle='--')
16 plt.show()
```

**Output:** Biểu đồ cho thấy biên độ của sóng âm thanh theo thời gian.

### speech recognition

- Cài đặt thư viện:

```
1 !pip install SpeechRecognition
```

- Sử dụng thư viện SpeechRecognition để chuyển âm thanh thành văn bản:

```
1 import speech_recognition as sr
2
3 r = sr.Recognizer()
4 audio_filename = 'data.wav'
5
6 my_audio = sr.AudioFile(audio_filename)
7 with my_audio as source:
8     audio = r.record(source)
9
10 print(type(audio))
11 your_speech = r.recognize_google(audio, language="vi-VN")
12 print("Audio transcription: ", your_speech)
```

**Output:** Hiện ra tận phía xa.

### gtts

- Cài đặt thư viện:

```
1 !pip install gtts
```

- Sử dụng thư viện SpeechRecognition để chuyển âm thanh thành văn bản:

```
1 from gtts import gTTS
2 import librosa
3 import numpy as np
4
5 lang='vi'
6 output_filename = 'record.mp3'
7 content = "xin chào mọi người"
8 output = gTTS(content, lang=lang, slow=False) # text to speech
9
10 output.save(output_filename) # save google audio to a file
11 data, sr = librosa.load(output_filename) # load google audio using
12     librosa library
13
14 import IPython
15 IPython.display.display(IPython.display.Audio(np.transpose(data),
16     rate=sr)) # display audio
```

**Output:** Một đoạn âm thanh mp3 với nội dung là "Xin chào mọi người".

### 3. Bài tập:

- Hãy thực hiện các thư viện trên với các input âm thanh và văn bản khác nhau để có thể hiểu rõ hơn về cách sử dụng các thư viện này.

```
1 test_case_1 = Một file âm thanh tự do.
2 test_case_2 = "Tôi yêu AIO"
3 # Your code here
```

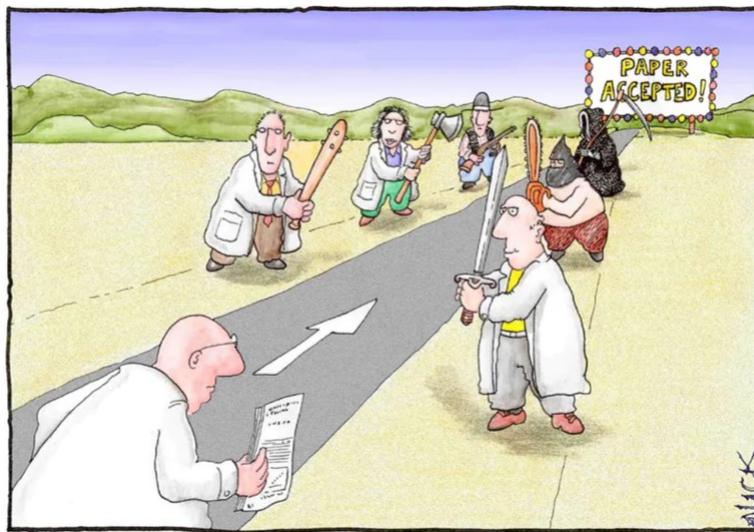
**Output:** Các đồ thị và đoạn văn bản tương ứng với file truyền vào.

**Output:** Một đoạn âm thanh mp3 với nội dung là "Tôi yêu AIO"

# Basic Python - Work with Text Data

Hoàng-Nguyễn Vũ

## 1. Mô tả: Làm quen với thư viện newspaper3k và NLTK



- **Thư viện newspaper3k** là một thư viện Python mã nguồn mở giúp bạn trích xuất dữ liệu từ các bài báo trực tuyến. Thư viện này hỗ trợ nhiều trang web tin tức khác nhau, bao gồm: VnExpress, Tuổi Trẻ, Thanh Niên, Zing News, VTC News, ... Các tính năng nổi bật của thư viện bao gồm:
  - + **Trích xuất dữ liệu:** Newspaper3k có thể trích xuất nhiều loại dữ liệu từ các trang web báo chí, bao gồm tiêu đề, bài viết, tóm tắt, tác giả, ngày tháng, hình ảnh, video, v.v.
  - + **Hỗ trợ nhiều trang web:** Newspaper3k hỗ trợ hơn 100 trang web báo chí khác nhau, bao gồm cả các trang web tiếng Việt như VnExpress, Tuổi Trẻ, Thanh Niên, v.v.
  - + **Dễ sử dụng:** Newspaper3k cung cấp một API đơn giản để trích xuất dữ liệu từ các trang web báo chí.
  - + **Mã nguồn mở:** Newspaper3k là một thư viện mã nguồn mở, vì vậy bạn có thể sử dụng và sửa đổi nó miễn phí.
- **Thư viện nltk (Natural Language Toolkit)** là một thư viện mã nguồn mở được phát triển bởi Python. Nó cung cấp một bộ công cụ mạnh mẽ để xử lý ngôn ngữ tự nhiên (NLP) trong Python. Các tính năng chính của thư viện bao gồm:
  - + **Phân tích cú pháp:** NLTK có thể phân tích cú pháp của các câu tiếng Anh để xác định cấu trúc và thành phần của chúng.
  - + **Phân loại từ:** NLTK có thể xác định loại từ (danh từ, động từ, tính từ, v.v.) của các từ trong một câu.

- + **Gán nhãn ngữ nghĩa:** NLTK có thể gán nhãn ngữ nghĩa (tên riêng, địa điểm, tổ chức, v.v.) cho các từ trong một câu.
- + **Tóm tắt văn bản:** NLTK có thể tóm tắt các văn bản dài thành các văn bản ngắn hơn.
- + **Dịch máy:** NLTK có thể dịch văn bản từ ngôn ngữ này sang ngôn ngữ khác.

- **Cách cài đặt và sử dụng một số tính năng:**

- Để cài đặt thư viện newspaper3k và NLTK, ta sẽ cài thông qua câu lệnh:

```
1 !pip install newspaper3k
2 !pip install nltk
```

- Cách sử dụng các tính năng chính của thư viện:

- + **Thư viện newspaper3k:**

```
1 from newspaper import Article
2
3 # Tạo một đối tượng Article từ URL của bài báo
4 article = Article('https://vnexpress.net/thoi-tiet-mien-bac-
    -mien-trung-mien-nam-ngay-14-3-4518045.html')
5
6 # Tải bài báo
7 article.download()
8 article.parse()
9
10 # In bài báo
11 print(article.text)
12
13 # Lấy toàn bộ ảnh trong bài báo
14 print(article.images)
```

+ **Kết quả:** Tòa án quân sự Mỹ thông báo Ryan Mays, thủy thủ bị tố đốt tàu đổ bộ USS Bonhomme Richard, được trả án....

{Link đường dẫn ảnh của bài báo...}

- + **Thư viện nltk:**

```
1 import nltk
2 from nltk.tokenize import word_tokenize
3
4 nltk.download('punkt')
5
6 data = "Tôi thích học AI và Toán"
7 # Bước 1: Tokenization data
8 tokenization = word_tokenize(data)
9 # Bước 2: Gọi thư viện Pos tagging
10 result = nltk.pos_tag(tokenization)
11 print(result)
```

+ **Kết quả:** [('Tôi', 'NNP'), ('thích', 'NN'), ('học', 'NN'), ('AI', 'NNP'), ('và', 'NN'), ('Toán', 'NNP')]

## 2. Bài tập:

- **Câu 1:** Thực hiện đọc và tóm tắt bài báo tại đường dẫn sau: Bài báo VnExpress
- **Câu 2:** Thực hiện pos tagging bài báo trên với thư viện NLTK.

### Kết quả:

+ **Câu 1:** Ngày 13/3, Cognition Labs, startup về công nghệ trí tuệ nhân tạo tại Mỹ, công bố kỹ sư phát triển phần mềm AI đầu tiên trên thế giới. Với Devin, các kỹ sư có thể tập trung vào những vấn đề thú vị hơn, các đội kỹ thuật có thể nỗ lực cho những mục tiêu tham vọng hơn", Cognition cho biết...

+ **Câu 2:** [('Kỹ', 'NNP'), ('sư', 'NN'), ('phần', 'NN'), ('mềm', 'NN'), ... ]

# Basic Python - Data Analysis with Visualization

*Hoàng-Nguyễn Vũ*

## 1. Mô tả: Làm quen với thư viện PygWalker



- **Thư viện PygWalker** là một thư viện Python mã nguồn mở giúp bạn dễ dàng chuyển đổi dữ liệu thành các ứng dụng phân tích trực quan. Thư viện này cung cấp một bộ công cụ mạnh mẽ để khám phá, tóm tắt và trực quan hóa dữ liệu của bạn, giúp bạn hiểu rõ hơn về dữ liệu và đưa ra quyết định sáng suốt hơn.
- **Điểm nổi bật của PygWalker:**
  - + **Tạo bảng điều khiển tương tác:** PygWalker cho phép bạn tạo các bảng điều khiển trực quan và dễ sử dụng để khám phá dữ liệu của bạn. Bạn có thể dễ dàng thêm và loại bỏ các biểu đồ, thay đổi bộ lọc và tương tác với dữ liệu theo thời gian thực.
  - + **Hỗ trợ nhiều loại biểu đồ:** PygWalker cung cấp nhiều loại biểu đồ khác nhau để trực quan hóa dữ liệu của bạn, bao gồm biểu đồ thanh, biểu đồ đường, biểu đồ phân tán, biểu đồ nhiệt, v.v.
  - + **Khả năng lọc và nhóm dữ liệu:** PygWalker cho phép bạn lọc dữ liệu theo các tiêu chí cụ thể và nhóm dữ liệu theo các trường khác nhau.
  - + **Tích hợp với Jupyter Notebook:** PygWalker có thể được sử dụng trong Jupyter Notebook, cho phép bạn kết hợp phân tích dữ liệu với mã Python khác.
  - + **Dễ sử dụng:** PygWalker có API đơn giản và dễ sử dụng, giúp bạn dễ dàng bắt đầu.
- **Ứng dụng của PygWalker trong việc trực quan hóa dữ liệu:**
  - + **Khoa học dữ liệu:** PygWalker có thể được sử dụng để khám phá và phân tích dữ liệu trong khoa học dữ liệu.
  - + **Học máy:** PygWalker có thể được sử dụng để chuẩn bị dữ liệu và đánh giá mô hình học máy.
  - + **Tài chính:** PygWalker có thể được sử dụng để phân tích dữ liệu tài chính và thị trường chứng khoán.
  - + **Tiếp thị:** PygWalker có thể được sử dụng để phân tích dữ liệu khách hàng và chiến dịch tiếp thị.

## 2. Cách cài đặt và sử dụng một số tính năng:

Để cài đặt thư viện PygWalker, chúng ta có thể cài trên Google Colab, hoặc ở máy cá nhân thông qua Jupyter Notebook. Cách cài đặt như sau:

### 1. Cài đặt thư viện PygWalker:

+ Để cài đặt thư PygWalker, chúng ta sử dụng câu lệnh sau ở Google Colab:

```
1 !pip install pygwalker
```

+ Để cài thư viện trên máy cá nhân và chạy với Jupyter Notebook, chúng ta sẽ chạy thông qua Terminal đối với hệ điều hành MacOS và CMD đối với hệ điều hành Windows thông qua lệnh sau:

```
1 pip install pandas
2 pip install pygwalker
```

```
[nguyen@Nguyen ~ % pip install pygwalker
Collecting pygwalker
  Downloading pygwalker-0.4.7-py3-none-any.whl.metadata (19 kB)
Collecting appdirs (from pygwalker)
  Downloading appdirs-1.4.4-py2.py3-none-any.whl.metadata (9.0 kB)
Requirement already satisfied: arrow in /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
```

Hình 1: Cài đặt PygWalker

+ Sau khi cài xong chúng ta khởi động jupyter notebook tại thư mục chứa project của chúng ta để sử dụng, thông qua lệnh sau:

```
1 jupyter notebook
```

```
nguyen@Nguyen Demo PygWalker % jupyter notebook
[I 2024-03-25 21:18:57.415 ServerApp] Package notebook took
[I 2024-03-25 21:18:57.457 ServerApp] Package jupyter_lsp took
[W 2024-03-25 21:18:57.457 ServerApp] A `__jupyter_server_extension` was not found in jupyter_lsp. Instead, a `__jupyter_server_extension` was found and will be used for now. This function will be removed in future releases of Jupyter Server.
```

Hình 2: Khởi chạy Jupyter Notebook

+ Để khởi động thư viện PygWalker trong Colab/Jupyter Notebook, trước tiên ta cần phải có dataset để thư viện có thể trực quan hóa dữ liệu. Sau khi chúng ta đã có data, để khởi tạo thư viện PygWalker như sau:

```
1 import pygwalker as pyg
2 import pandas as pd
3 # FILE_PATH: đường dẫn tới tập tin CSV
4 data = pd.read_csv(FILE_PATH)
5 pyg.walk(data)
```

Showing 1 to 100 of 199 results

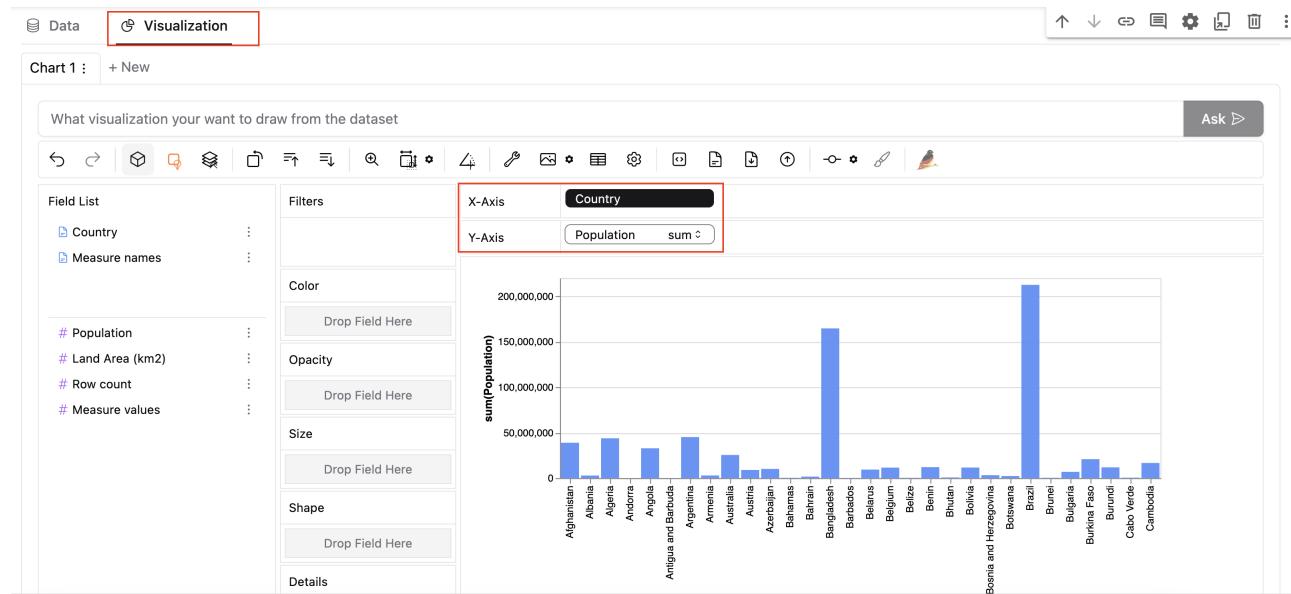
Country	Population	Land Area (km2)
<b>199</b> unique values	<b>199</b> unique values	700 1% 460 1% Other (195) 98%
Brunei	212,559,417	5,270,140
Brunei	437,479	5,270
Bulgaria	6,948,445	108,560
Burkina Faso	20,903,273	273,600
Burundi	11,890,784	25,680
Cabo Verde	555,987	4,030
Cambodia	16,718,965	176,520
Cameroon	26,545,863	472,710
Canada	37,742,154	9,093,510
Central African Republic	4,829,767	622,980
Chad	16,425,864	1,259,200
Chile	19,116,201	743,532
China	1,439,323,776	9,388,211
Colombia	50,882,891	1,109,500

Hình 3: Giao diện PygWalker

## 2. Sử dụng một số tính năng trực quan hóa:

- + **Tạo biểu đồ cột:** Tạo biểu đồ cột cho tập data mẫu trên để thể hiện dân số (Population) theo quốc gia (Country).

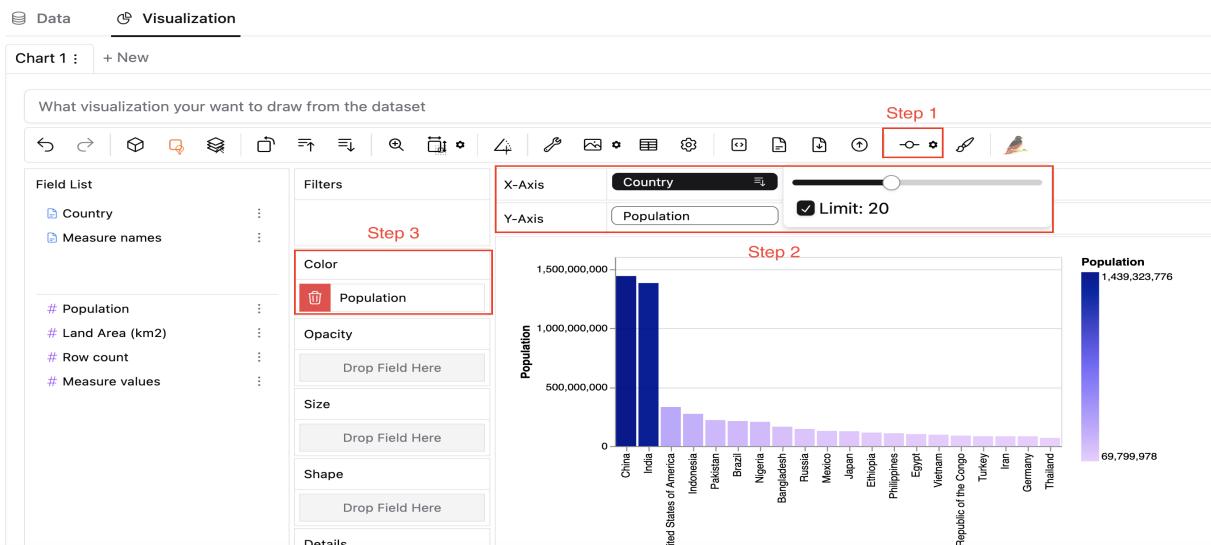
(\*) Ta sẽ thực hiện kéo 2 cột: Country vào X-Axis và Population vào Y-Axis. Ứng với 2 thông số của biểu đồ cột mà chúng ta cần thực hiện trực quan hóa biểu đồ: Trục Ox (Trục ngang) thể hiện cho Quốc gia (Country) và Trục Oy (Trục dọc) thể hiện cho Dân số (Population)



Hình 4: Biểu đồ cột thể hiện dân số theo quốc gia

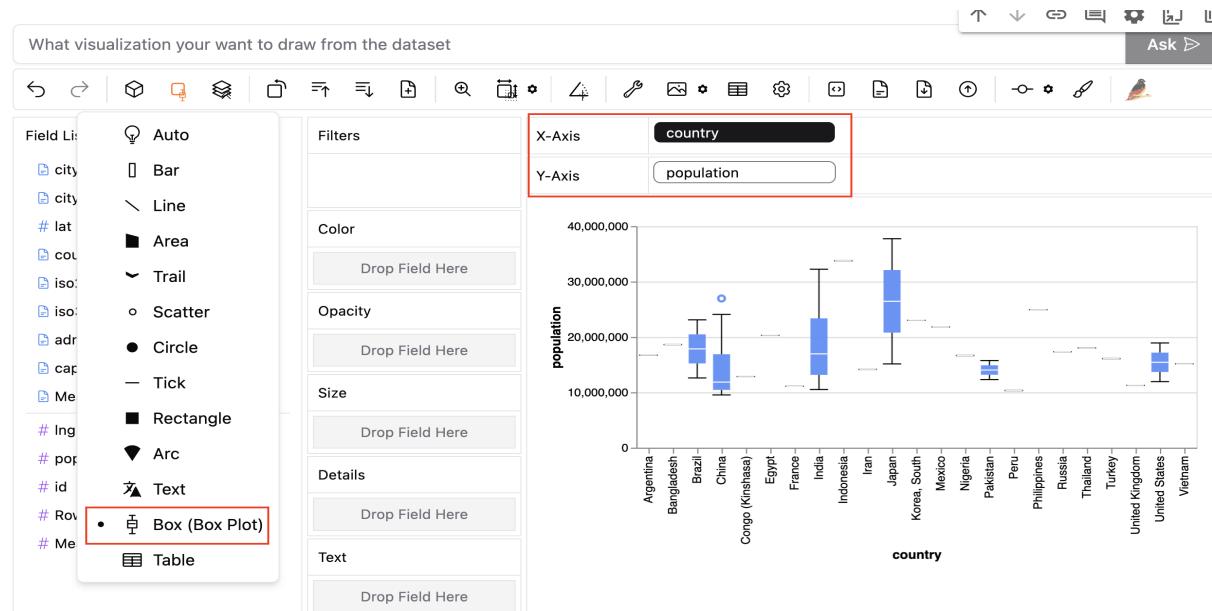
+ **Lấy Top 20 Quốc Gia có giảm dần theo dân số, và tô màu theo độ lớn của dân số:**

(\*) Ta sẽ thực hiện kéo 2 cột: Chúng ta cũng thực hiện tương tự bài trên nhưng chúng sẽ thực hiện sắp xếp Population giảm dần và Limit 20 dòng. Đồng thời gắn Color là cột Population để thư viện hiện hiện tô màu theo Population.



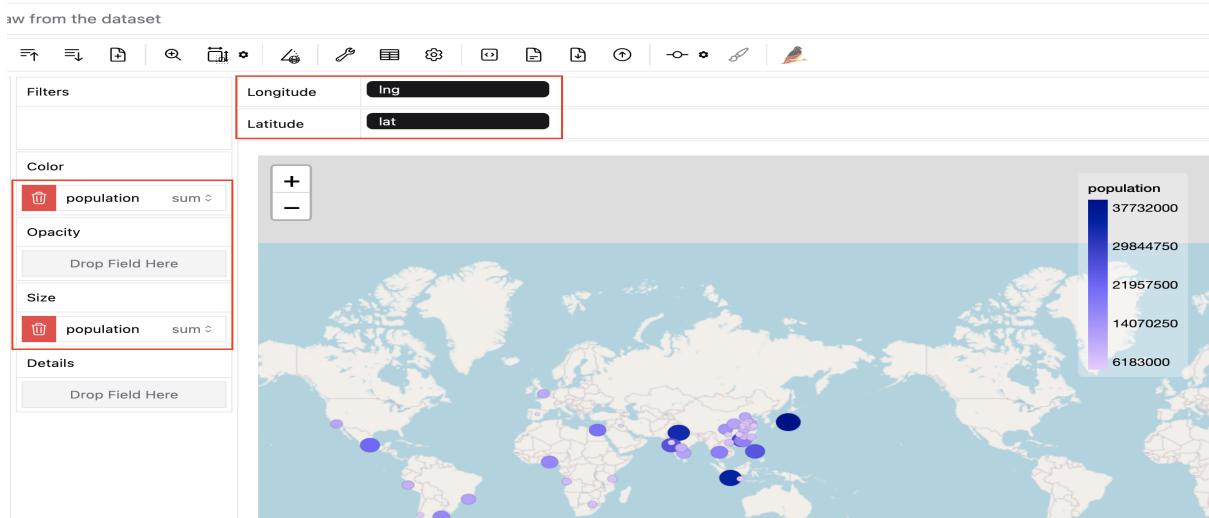
Hình 5: Biểu đồ cột thể hiện dân số theo quốc gia

+ **Vẽ biểu đồ hộp thể hiện phân phối dữ liệu:**



Hình 6: Biểu đồ hộp thể hiện dân số theo quốc gia

+ Vẽ bản đồ hộp thể hiện phân phối dữ liệu:

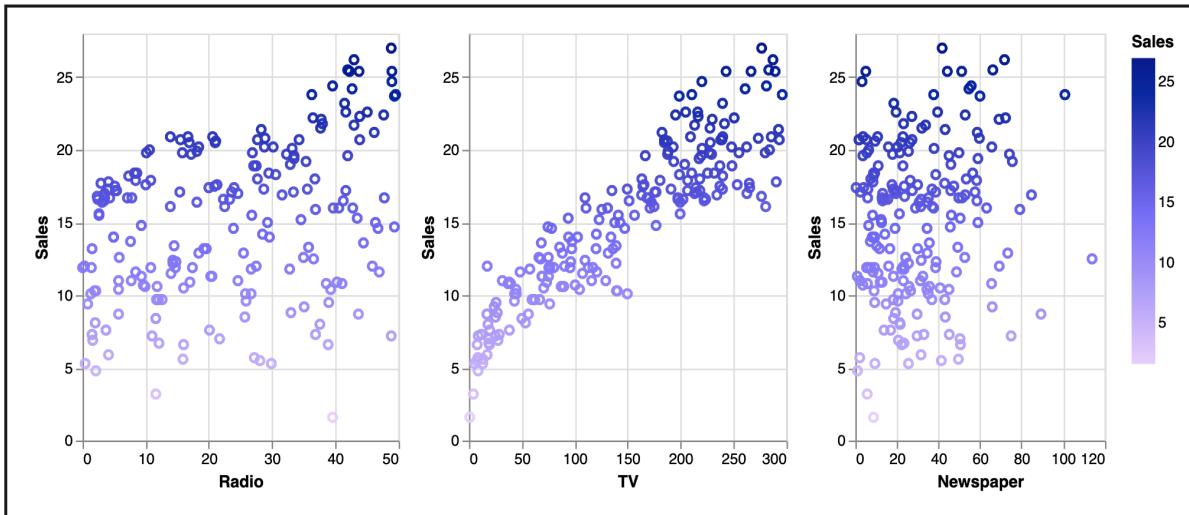


Hình 7: Bản đồ thể hiện dân số theo quốc gia

3. **Bài tập:** Hãy đọc dữ liệu ở file: advertising.csv và khởi chạy thư viện PygWalker sau đó thực hiện trực quan các biểu đồ sau đây:

- **Câu 1:** Vẽ biểu đồ phân phối dữ liệu cho 3 loại: TV, Radio, Paper và được Color theo độ lớn của giá bán (Sales).

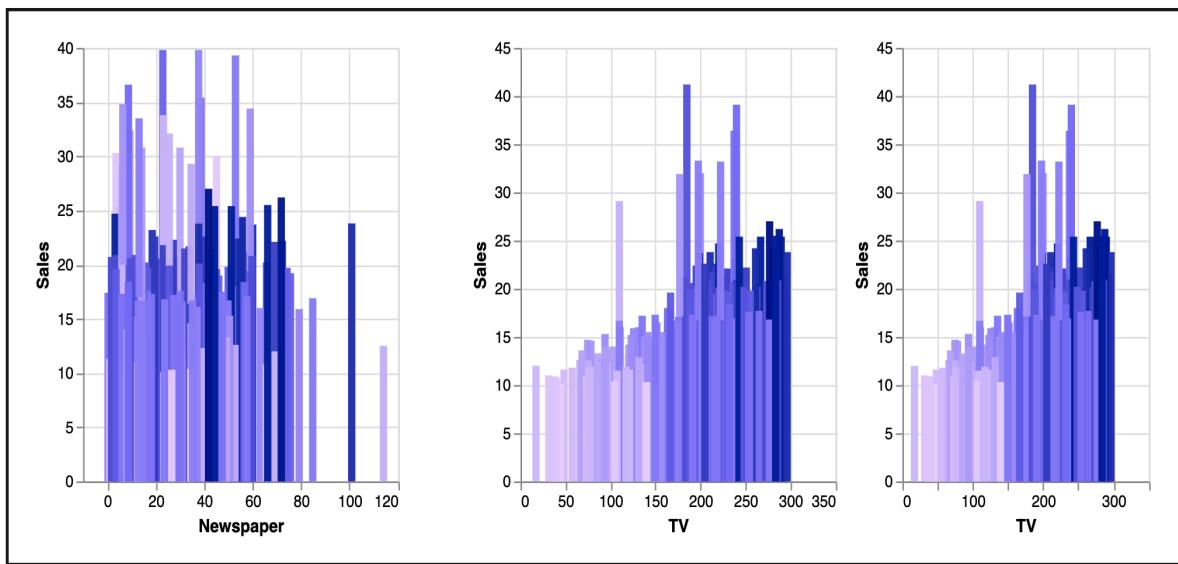
### Kết Quả:



Hình 8: Biểu đồ phân phối dữ liệu cho 3 loại: TV, Radio, Paper

- **Câu 2:** Vẽ biểu đồ cột thể hiện doanh số bán (Sales)  $\geq 10$  của cả 3 loại TV, Radio và Newspaper.

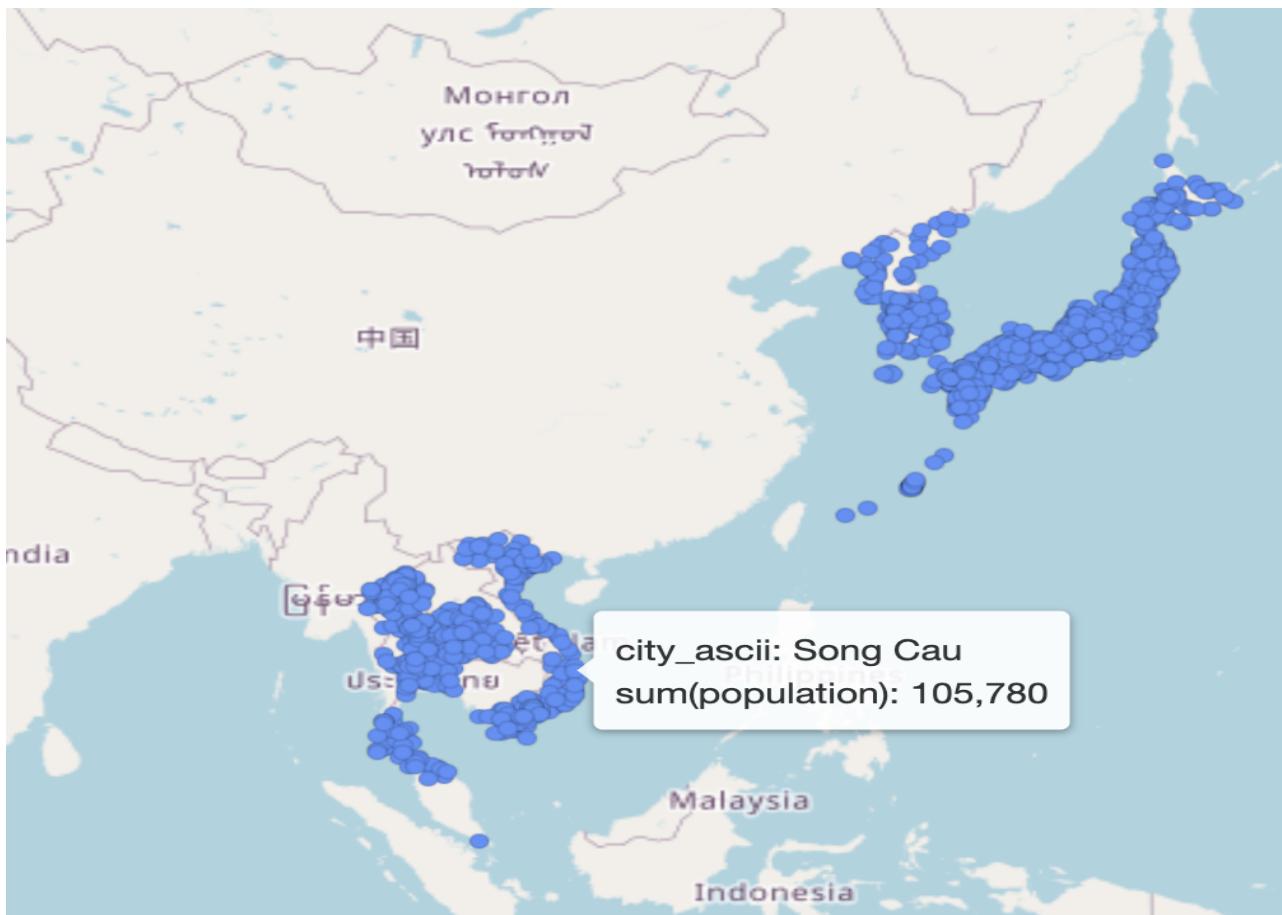
### Kết quả:



Hình 9: Biểu đồ doanh số bán hàng trên 10 sản phẩm cho 3 loại: TV, Radio, Paper

- **Câu 3:** Hãy vẽ bản đồ biểu diễn phân bố dân số theo thành phố của các nước thuộc các nước: Việt Nam, Hàn Quốc, Nhật Bản, Singapore và Thái Lan dựa theo dữ liệu sau: Dữ liệu dân số thế giới

Kết quả:



Hình 10: Bản đồ thể hiện phân bố dân số theo thành phố của các nước thuộc các nước: Việt Nam, Hàn Quốc, Nhật Bản, Singapore và Thái Lan

- Hết -

# Basic Python - Work with Text Data

*Hoàng-Nguyễn Vũ*

## 1. Mô tả: Làm quen với thư viện translate và googletrans

- **Thư viện Translate** là một thư viện mã nguồn mở được phát triển bởi Google. Nó cho phép bạn dịch văn bản, mã, tài liệu và trang web sang nhiều ngôn ngữ khác nhau. Thư viện này sử dụng công nghệ dịch máy của Google, được đào tạo trên một lượng lớn dữ liệu ngôn ngữ.
- **Googletrans** là một thư viện Python dựa trên thư viện Translate. Nó cung cấp một API đơn giản để dịch văn bản sang nhiều ngôn ngữ khác nhau. Googletrans cũng sử dụng công nghệ dịch máy của Google.
- **Điểm khác biệt chính giữa hai thư viện:**

Tính năng	Thư viện Translate	Googletrans
Ngôn ngữ hỗ trợ	Nhiều hơn	Ít hơn
Loại dữ liệu	Văn bản, mã, tài liệu, trang web	Văn bản
API	Phức tạp	Đơn giản
Dễ sử dụng	Khó hơn	Dễ hơn

### • Cách cài đặt và sử dụng một số tính năng:

- Để cài đặt thư viện translate và Googletrans, ta sẽ cài thông qua câu lệnh:

```
1 !pip install translate
2 !pip install googletrans==4.0.0rc1
```

- Cách sử dụng dịch thuật trong thư viện translate và Googletrans:

+ **Thư viện Translate:**

```
1 from translate import Translator
2
3 translator= Translator(to_lang="vi")
4 translation = translator.translate("Don't cry because it's
      over, smile because it happened.")
5 print(translation)
```

+ **Kết quả:** Đừng khóc vì nó đã kết thúc, hãy mỉm cười vì nó đã xảy ra.

+ **Thư viện Googletrans:** Dịch từ tiếng Việt sang tiếng Pháp

```
1 from googletrans import Translator
2
3 # define a translate object
4 translate = Translator()
5
6 # Translate some text
```

```
7 result = translate.translate('Hôm nay thời tiết không tốt v  
à mưa nhiều', dest='fr')  
8  
9 print(result)  
10 print(result.text)
```

+ **Kết quả:** Translated(src=vi, dest=fr, text=Aujourd’hui, le temps n'est pas bon et pluvieux, pronunciation=None, extra\_data="'confiden...")  
Aujourd’hui, le temps n'est pas bon et pluvieux

## 2. Bài tập:

- Hãy thực hiện dịch câu sau với 2 ngôn ngữ: Anh, Nhật sử dụng cả 2 thư viện:

Công cụ Suno AI nhanh chóng nhận được sự chú ý từ người dùng khi có thể tạo bài hát chỉ với vài câu lệnh. Phiên bản mới nhất V3 Alpha mới được giới thiệu cuối tháng 2, có bản miễn phí với 10 bài hát mỗi ngày.

### Kết quả:

+ **Tiếng Anh:** Suno AI tools quickly get attention from users when they can create a song with just a few statements. The latest version of V3 Alpha was introduced at the end of February, with a free version with 10 songs a day.

+ **Tiếng Nhật:** Suno AIツールは、ユーザーがいくつかのステートメントで曲を作成できる場合、ユーザーからすぐに注目を集めます。V3 Alphaの最新バージョンは2月末に紹介され、1日10曲の無料バージョンがありました。

# Basic Python - Work with Text Data

*Hoàng-Nguyễn Vũ*

## 1. Mô tả: Làm quen với thư viện Underthesea

- **Underthesea** là một bộ công cụ mã nguồn mở hỗ trợ xử lý ngôn ngữ tự nhiên Tiếng Việt (NLP). Nó được phát triển bởi cộng đồng nghiên cứu NLP tại Việt Nam và được công bố lần đầu tiên vào năm 2017.

Bảng 1: Tính năng, ưu điểm của thư viện Underthesea

Tính năng	
Phân chia câu	Cắt một đoạn văn bản thành các câu riêng biệt.
Phân loại từ	Xác định loại từ (danh từ, động từ, tính từ, v.v.) của mỗi từ trong câu.
Gán thẻ POS	Gán thẻ cho mỗi từ với thông tin ngữ pháp (danh từ, động từ, tính từ, v.v.).
Nhận dạng thực thể tên riêng	Xác định các thực thể tên riêng (người, địa điểm, tổ chức, v.v.) trong văn bản.
Phân loại văn bản	Phân loại văn bản vào các chủ đề hoặc thể loại khác nhau.
Tóm tắt văn bản	Tạo tóm tắt ngắn gọn cho một đoạn văn bản dài.
Trích xuất quan điểm	Xác định các quan điểm và ý kiến trong văn bản.
Dịch máy	Dịch văn bản từ tiếng Việt sang tiếng Anh và ngược lại.
Ưu điểm	
Mã nguồn mở	Có thể sử dụng và sửa đổi miễn phí.
Dễ sử dụng	Cung cấp API đơn giản và dễ hiểu.
Hiệu quả	Đã được chứng minh hiệu quả trên nhiều tập dữ liệu tiếng Việt.
Cộng đồng	Được hỗ trợ bởi cộng đồng nghiên cứu NLP Việt Nam năng động.

### • Cách cài đặt và sử dụng một số tính năng:

- Để cài đặt thư viện Underthesea, ta sẽ cài thông qua câu lệnh:

```
1 !pip install underthesea
```

- Các tính năng nổi bật trong thư viện Underthesea:

- + **Gán nhãn từ loại (POS tagging):**

```
1 from underthesea import pos_tag
2 pos_tag('Học sinh đang học toán')
```

+ Kết quả: [('Học sinh', 'N'), ('đang', 'R'), ('học', 'V'), ('toán', 'N')]

+ Phân loại văn bản (Text classification):

```
1 from underthesea import classify
2 classify('giá cổ phiếu đang có nhiều biến động trong thời gian qua')
```

+ Kết quả: ['kinh\_doanh']

+ Phân tích cảm xúc (Sentiment Analysis):

```
1 from underthesea import sentiment
2 sentiment('Sản phẩm mình đặt về không như quảng cáo')
```

+ Kết quả: 'negative'

+ Phân đoạn câu văn (Sentence Segmentation):

```
1 from underthesea import sent_tokenize
2 text = 'Những đứa trẻ nghèo thế hệ tôi biết đọc biết viết, thành người bằng những cuốn giáo khoa đi mượn như thế. Cũng có những đứa nhà nghèo quá, không mượn đâu được bộ sách cho tử tế, môn được môn không, càng học càng đúp, cuối cùng bỏ dở giữa chừng.'
3 sent_tokenize(text)
```

+ Kết quả: ['Những đứa trẻ nghèo thế hệ tôi biết đọc biết viết, thành người bằng những cuốn giáo khoa đi mượn như thế.', 'Cũng có những đứa nhà nghèo quá, không mượn đâu được bộ sách cho tử tế, môn được môn không, càng học càng đúp, cuối cùng bỏ dở giữa chừng. ]

+ Phân đoạn từ ngữ (Word Segmentation):

```
1 from underthesea import word_tokenize
2 sentence = 'Công trình của PGS Vân đã thay thế bạch kim trong pin nhiên liệu, giúp giảm giá thành mà pin có độ bền cao hơn.'
3 word_tokenize(sentence)
```

+ Kết quả: ['Công', 'trình', 'của', 'PGS', 'Vân', 'đã', 'thay', 'thế', 'bạch', 'kim', 'trong', 'pin', 'nhiên', 'liệu', ',', 'giúp', 'giảm', 'giá', 'thành', 'mà', 'pin', 'có', 'độ', 'bền', 'cao', 'hơn', ':']

## 2. Bài tập:

- Hãy thực hiện các task: POS Tagging, Text Classification, Sentiment Analysis, Sentence Segmentation, Word Segmentation đoạn văn bản sau:

Công cụ Suno AI nhanh chóng nhận được sự chú ý từ người dùng khi có thể tạo bài hát chỉ với vài câu lệnh. Phiên bản mới nhất V3 Alpha mới được giới thiệu cuối tháng 2, có bản miễn phí với 10 bài hát mỗi ngày.

Kết quả:

- + **POS Tagging:** [('Công cụ', 'N'), ('Suno', 'Np'), ('AI', 'P'), ... ]
- + **Text Classification:** ['vi\_tinh']
- + **Sentiment Analysis:** 'positive'
- + **Sentence Segmentation:** ['Công cụ Suno AI nhanh chóng nhận được sự chú ý từ người dùng khi có thể tạo bài hát chỉ với vài câu lệnh.', 'Phiên bản mới nhất V3 Alpha mới được giới thiệu cuối tháng 2, có bản miễn phí với 10 bài hát mỗi ngày.]
- + **Word Segmentation:** ['Công cụ', 'Suno', 'AI', 'nhanh chóng', ...]

# Basic Python - Basic Data Analysis

Hoàng-Nguyễn Vũ

## 1 Correlation (Tương quan)

Correlation là một phép đo thống kê cho biết mức độ và hướng của mối quan hệ giữa hai biến số. Khi hai biến số có mối quan hệ tương quan, giá trị của một biến có thể ảnh hưởng đến giá trị của biến còn lại.

- **Positive Correlation (Tương quan dương):** Khi giá trị của một biến tăng thì giá trị của biến kia cũng tăng.
- **Negative Correlation (Tương quan âm):** Khi giá trị của một biến tăng thì giá trị của biến kia giảm.
- **No Correlation (Không tương quan):** Không có mối quan hệ rõ ràng giữa hai biến.

## 2 Correlation Coefficient (Hệ số tương quan)

Hệ số tương quan là một giá trị số từ -1 đến 1 cho biết mức độ của mối quan hệ tương quan:

- **1:** Mối quan hệ tương quan hoàn hảo dương.
- **0:** Không có mối quan hệ tương quan.
- **-1:** Mối quan hệ tương quan hoàn hảo âm.

### Công thức tính hệ số tương quan Pearson (Pearson Correlation Coefficient)

$$r = \frac{\sum (X - \bar{X})(Y - \bar{Y})}{\sqrt{\sum (X - \bar{X})^2 \sum (Y - \bar{Y})^2}}$$

Trong đó:

- $X$  và  $Y$  là hai biến số.
- $\bar{X}$  và  $\bar{Y}$  là giá trị trung bình của  $X$  và  $Y$ .

## 3 Bài tập: Tính hệ số tương quan giữa các feature trong tập dữ liệu advertise

1. Đọc file dữ liệu advertising, tại đây.
2. Hãy tính hệ số tương quan giữa các features có trong tập dataset này.

## Hướng dẫn:

```

1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5
6 # Đọc dataset
7 data = pd.read_csv("advertising.csv")
8
9 def correlation(x, y):
10     ## Your code here ##
11
12 # Example usage:
13 x = data['TV']
14 y = data['Radio']
15 corr_xy = correlation(x, y)
16 print(f"Correlation between TV and Sales: {corr_xy}") ## Output: 0.0548

```

### 3.1 Bài tập mở rộng

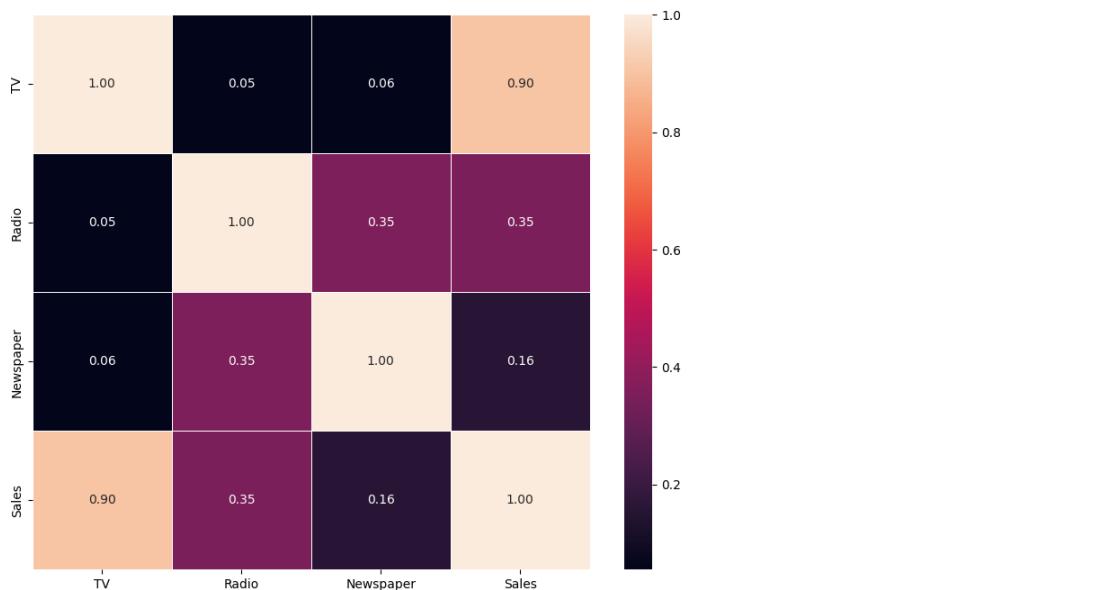
Từ giá trị correlation coefficient ở trên, hãy vẽ biểu đồ heatmap để biểu hiện sự tương quan.

## Hướng dẫn:

```

1 ## Calculate coerelation here ##
2 plt.figure(figsize=(10,8))
3 sns.heatmap(#Your result coerelation here#, annot=True, fmt=".2f",
4             linewidth=.5)
5 plt.show()

```



# Basic Linear Algebra with Numpy

*Hoàng-Nguyễn Vũ*

## 1 Giới Thiệu về NumPy

NumPy (**Numerical Python**) là một thư viện mạnh mẽ trong Python được sử dụng để xử lý dữ liệu số, đặc biệt là các phép toán trên **mảng đa chiều** (NumPy array) và **các phép toán đại số tuyến tính**. NumPy giúp tối ưu tốc độ tính toán, hỗ trợ nhiều thuật toán khoa học, Machine Learning và xử lý dữ liệu.

### 1.1 Cài Đặt NumPy

Bạn có thể cài đặt NumPy bằng lệnh:

```
1 pip install numpy
```

### 1.2 Khởi Tạo Mảng NumPy

```
1 import numpy as np
2
3 # Tạo mảng 1D (vector)
4 a = np.array([1, 2, 3, 4])
5
6 # Tạo mảng 2D (ma trận)
7 b = np.array([[1, 2], [3, 4]])
8
9 # Tạo mảng số 0, số 1 và mảng ngẫu nhiên
10 zero_matrix = np.zeros((3,3))
11 ones_matrix = np.ones((2,2))
12 rand_matrix = np.random.rand(3,3)
13
14 # Tạo ma trận đơn vị
15 identity_matrix = np.eye(3)
```

## 2 Các Phép Toán Cơ Bản với NumPy

### 2.1 Phép Cộng, Trừ Hai Ma Trận

Cho hai ma trận  $A$  và  $B$  cùng kích thước  $m \times n$ :

$$C = A + B, \quad D = A - B$$

Công thức tổng quát:

$$c_{ij} = a_{ij} + b_{ij}, \quad d_{ij} = a_{ij} - b_{ij}, \quad \forall i, j$$

Ví dụ trong NumPy:

```

1 A = np.array([[1, 2], [3, 4]])
2 B = np.array([[5, 6], [7, 8]])
3
4 C = A + B # Cộng ma trận
5 D = A - B # Trừ ma trận

```

## 2.2 Phép Nhân Từng Phần Tử (Element-wise)

Nhân từng phần tử của hai ma trận:

$$E = A \odot B, \quad e_{ij} = a_{ij} \cdot b_{ij}$$

```

1 E = A * B # Nhân từng phần tử

```

## 2.3 Phép Nhân Ma Trận (Matrix Multiplication)

$$C = A \times B, \quad c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$$

Ví dụ trong NumPy:

```

1 C = A @ B # Hoặc np.dot(A, B)

```

# 3 Các Phép Toán Đại Số Tuyến Tính trong NumPy

## 3.1 Tính Định Thức Ma Trận

$$\det(A) = a_{11}a_{22} - a_{12}a_{21}$$

Ví dụ trong NumPy:

```

1 det_A = np.linalg.det(A)

```

## 3.2 Ma Trận Nghịch Đảo

Một ma trận  $A$  khả nghịch nếu tồn tại  $A^{-1}$  sao cho:

$$AA^{-1} = A^{-1}A = I$$

Công thức tính:

$$A^{-1} = \frac{1}{\det(A)} \cdot \text{adj}(A)$$

Ví dụ trong NumPy:

```

1 A_inv = np.linalg.inv(A)

```

### 3.3 Hạng của Ma Trận

Hạng của ma trận là số hàng độc lập tuyến tính:

```
1 rank_A = np.linalg.matrix_rank(A)
```

### 3.4 Giải Hệ Phương Trình Tuyến Tính

Giải hệ phương trình  $Ax = B$ :

$$x = A^{-1}B$$

Ví dụ trong NumPy:

```
1 x = np.linalg.solve(A, B)
```

### 3.5 Tính Giá Trị Riêng và Vector Riêng

$$Av = \lambda v$$

Giải phương trình đặc trưng:

$$\det(A - \lambda I) = 0$$

Ví dụ trong NumPy:

```
1 eig_values, eig_vectors = np.linalg.eig(A)
```

### 3.6 Phân Rã SVD (Singular Value Decomposition)

$$A = USV^T$$

Ví dụ trong NumPy:

```
1 U, S, V = np.linalg.svd(A)
```

## 4 Các Phép Toán Vector

### 4.1 Tích Vô Hướng (Dot Product)

$$v_1 \cdot v_2 = \sum_{i=1}^n v_{1i}v_{2i}$$

Ví dụ trong NumPy:

```
1 dot_product = np.dot(v1, v2)
```

## 4.2 Tích Có Hướng (Cross Product)

$$v_1 \times v_2 = \begin{vmatrix} i & j & k \\ v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \end{vmatrix}$$

Ví dụ trong NumPy:

```
1 cross_product = np.cross(v1, v2)
```

## 4.3 Chuẩn của Vector (Norm)

Chuẩn Euclidean:

$$\|v\| = \sqrt{\sum_i v_i^2}$$

Ví dụ trong NumPy:

```
1 norm_v = np.linalg.norm(v)
```

## 4.4 Chuyển Vị Ma Trận

$$A^T = \text{hoán đổi hàng và cột của } A$$

Ví dụ trong NumPy:

```
1 A_transpose = A.T
```

## 5 Kết Luận

NumPy là một thư viện mạnh mẽ cho tính toán khoa học, giúp xử lý các phép toán đại số tuyến tính nhanh chóng. Việc hiểu rõ NumPy giúp ứng dụng hiệu quả vào Machine Learning, AI, và xử lý dữ liệu.

## Bài tập: Hãy viết chương trình Python sử dụng Numpy để thực hiện các phép toán

### Phép Cộng và Trừ Ma Trận

**Bài toán 1:** Cho hai ma trận:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

Tính:

$$C = A + B, \quad D = A - B$$

Kết quả:

$$C = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}, \quad D = \begin{bmatrix} -4 & -4 \\ -4 & -4 \end{bmatrix}$$

### Bài toán 2: Tính tổng và hiệu của:

$$C = \begin{bmatrix} 2 & -3 \\ 4 & -5 \end{bmatrix}, \quad D = \begin{bmatrix} 1 & 4 \\ -2 & 6 \end{bmatrix}$$

Kết quả:

$$C + D = \begin{bmatrix} 3 & 1 \\ 2 & 1 \end{bmatrix}, \quad C - D = \begin{bmatrix} 1 & -7 \\ 6 & -11 \end{bmatrix}$$

### Phép Nhân Hadamard

### Bài toán 3: Tính nhân Hadamard của hai ma trận:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

Kết quả:

$$A \odot B = \begin{bmatrix} 5 & 12 \\ 21 & 32 \end{bmatrix}$$

### Bài toán 4: Cho ma trận như bên dưới hãy tính tích Hadamard $E \odot F$

$$E = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}, \quad F = \begin{bmatrix} 5 & 4 & 3 \\ 2 & 1 & 0 \end{bmatrix}$$

Kết quả:

$$E \odot F = \begin{bmatrix} 0 & 4 & 6 \\ 6 & 4 & 0 \end{bmatrix}$$

## Tích Vô Hướng (Dot Product)

**Bài toán 5:** Cho hai vector:

$$v_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \quad v_2 = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$$

Tính tích vô hướng:  $v_1 \cdot v_2$ .

Kết quả:

$$v_1 \cdot v_2 = 1(4) + 2(5) + 3(6) = 32$$

## Tích Chập (Convolution)

**Bài toán 6:** Cho ma trận đầu vào và bộ lọc:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad K = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Thực hiện tích chập  $C = A * K$ .

Kết quả:

$$C = \begin{bmatrix} -4 & -4 \\ -4 & -4 \end{bmatrix}$$

## Xoay Vector Một Góc $\theta$

**Bài toán 8:** Xoay vector  $v = [1, 0]$  một góc  $\theta = 45^\circ$  quanh gốc tọa độ.

Kết quả:

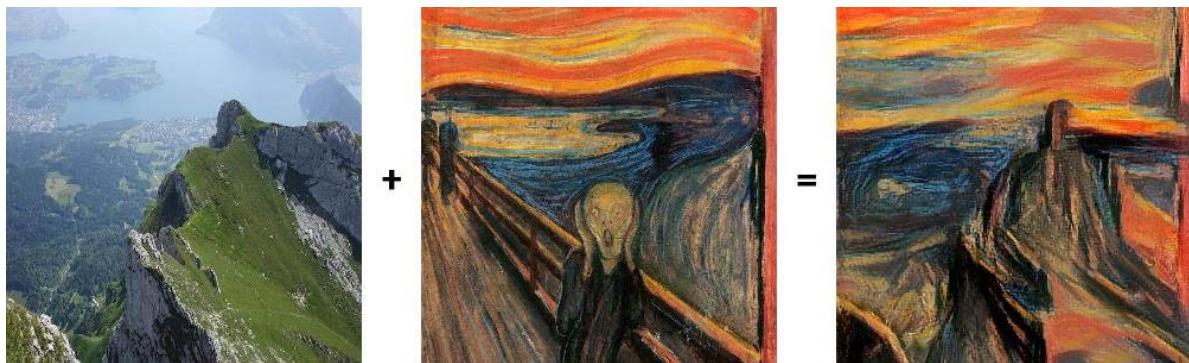
$$v_{\text{rotated}} = [0.52532199, -0.85090352]$$

# Numpy - Gram Matrix in Style transfer

*Hoàng-Nguyễn Vũ*

## 1 Giới thiệu

**Gram Matrix** là một công cụ quan trọng trong bài toán *Style Transfer*. Nó được sử dụng để trích xuất thông tin về phong cách của một hình ảnh dựa trên ma trận đặc trưng (*feature map*) của một mạng nơ-ron.



## 2 Vai trò của Gram Matrix trong Style Transfer

Gram Matrix giúp nắm bắt mối quan hệ tương quan giữa các feature maps và đại diện cho cấu trúc tổng thể của hình ảnh, chẳng hạn như các mẫu và kết cấu, chứ không phải các chi tiết cục bộ. Trong Style Transfer, Gram Matrix của hình ảnh phong cách được so sánh với Gram Matrix của hình ảnh kết quả để đảm bảo rằng phong cách của hình ảnh gốc được chuyển sang hình ảnh mới.

## 3 Công thức Gram Matrix

Giả sử chúng ta có một ma trận đặc trưng  $F$  của một ảnh, với kích thước:

$$F \in \mathbb{R}^{C \times H \times W}$$

trong đó:

- $C$  là số kênh (channels) của ảnh.
- $H$  là chiều cao (height) của ảnh.
- $W$  là chiều rộng (width) của ảnh.

Gram Matrix được tính theo công thức:

$$G = FF^T$$

trong đó:

- $G \in \mathbb{R}^{C \times C}$  là Gram Matrix.
- $F$  được reshape về kích thước  $(C, H \times W)$ .

Để tránh giá trị quá lớn, Gram Matrix thường được chuẩn hóa bằng số điểm ảnh:

$$G = \frac{FF^T}{H \times W}$$

## 4 Cài đặt bằng NumPy

Dưới đây là cách cài đặt Gram Matrix bằng NumPy:

```

1 import numpy as np
2
3 def compute_gram_matrix(feature_map: np.ndarray) -> np.ndarray:
4     """
5         Tính Gram Matrix từ feature map.
6
7     Args:
8         feature_map (np.ndarray): Ma trận đặc trưng có kích thước (C, H, W).
9
10    Returns:
11        np.ndarray: Gram Matrix có kích thước (C, C).
12    """
13    # Lấy kích thước đầu vào
14    C, H, W = feature_map.shape
15
16    # Chuyển đổi ma trận về dạng (C, H*W)
17    F = feature_map.reshape(C, H * W)
18
19    # Tính Gram Matrix G = F @ F.T
20    G = np.dot(F, F.T)
21
22    # Chuẩn hóa bằng số điểm ảnh
23    G /= (H * W)
24
25    return G
26
27 # Tạo dữ liệu giả lập với kích thước (3, 4, 4)
28 np.random.seed(42)
29 feature_map = np.random.rand(3, 4, 4)
30
31 # Tính Gram Matrix
32 gram_matrix = compute_gram_matrix(feature_map)

```

```
33  
34 # In kết quả  
35 print("Gram Matrix:\n", gram_matrix)
```

## 5 So sánh với PyTorch (Đọc thêm)

Có thể kiểm tra tính chính xác bằng cách so sánh với PyTorch:

```
1 import torch  
2  
3 feature_map_torch = torch.tensor(feature_map, dtype=torch.float32)  
4 F_torch = feature_map_torch.view(3, -1)  
5 gram_matrix_torch = torch.mm(F_torch, F_torch.t()) / (4 * 4)  
6  
7 print("\nGram Matrix PyTorch:\n", gram_matrix_torch.numpy())
```

## 6 Kết luận

Gram Matrix giúp mô hình Style Transfer học phong cách của một hình ảnh bằng cách đo tương quan giữa các kênh đặc trưng. Công thức này có thể được triển khai dễ dàng bằng NumPy và có thể kiểm tra với PyTorch để đảm bảo tính chính xác.

## 7 Bài tập

### Bài tập 1: Tính Gram Matrix từ Feature Map

#### Mô tả bài toán

Cho một ma trận đặc trưng  $F$  có kích thước:

$$F \in \mathbb{R}^{C \times H \times W}$$

trong đó:

- $C$  là số kênh (channels).
- $H$  là chiều cao của ảnh.
- $W$  là chiều rộng của ảnh.

Gram Matrix được tính theo công thức:

$$G = \frac{F \cdot F^T}{H \times W}$$

## Cài đặt bằng NumPy

Dưới đây là code Python để tính Gram Matrix từ feature map:

```

1 import numpy as np
2
3 def compute_gram_matrix(feature_map: np.ndarray) -> np.ndarray:
4     # Your code here #
5
6 # Feature Map đầu vào có định
7 feature_map = np.array([
8     [[1, 2], [3, 4]], # Kênh 1
9     [[5, 6], [7, 8]], # Kênh 2
10    [[9, 10], [11, 12]] # Kênh 3
11])
12
13 # Tính Gram Matrix
14 gram_matrix = compute_gram_matrix(feature_map)
15 print(gram_matrix)

```

## Kết quả mong muốn

Nếu bạn chạy đoạn code trên, kết quả đầu ra sẽ là:

$$G = \begin{bmatrix} 7.5 & 17.5 & 27.5 \\ 17.5 & 43.5 & 69.5 \\ 27.5 & 69.5 & 111.5 \end{bmatrix}$$

## Bài tập 2: Đo độ tương đồng giữa hai ảnh bằng Gram Matrix

### Mô tả bài toán

Cho hai ảnh với ma trận đặc trưng khác nhau, hãy đo **độ tương đồng** phong cách giữa chúng bằng Gram Matrix sử dụng công thức:

$$\text{Similarity} = \frac{\sum(G_1 \cdot G_2)}{\sqrt{\sum G_1^2} \cdot \sqrt{\sum G_2^2}}$$

trong đó:

- $G_1, G_2$  là Gram Matrix của hai ảnh.
- Giá trị đầu ra nằm trong khoảng  $[0,1]$ , càng gần 1 thì hai ảnh càng giống nhau về phong cách.

## Cài đặt bằng NumPy

```
1 def compute_similarity(gram1: np.ndarray, gram2: np.ndarray) -> float:
2     """
3         Tính độ tương đồng giữa hai Gram Matrix.
4
5     Args:
6         gram1 (np.ndarray): Gram Matrix ảnh 1.
7         gram2 (np.ndarray): Gram Matrix ảnh 2.
8
9     Returns:
10        float: Độ tương đồng trong khoảng [0,1].
11        """
12    # Your code here #
13
14 # Feature Map của hai ảnh
15 feature_map1 = np.array([
16     [[1, 2], [3, 4]],
17     [[5, 6], [7, 8]],
18     [[9, 10], [11, 12]]
19 ])
20
21 feature_map2 = np.array([
22     [[2, 4], [6, 8]],
23     [[1, 3], [5, 7]],
24     [[0, 2], [4, 6]]
25 ])
26
27 # Tính Gram Matrix của hai ảnh
28 gram1 = compute_gram_matrix(feature_map1)
29 gram2 = compute_gram_matrix(feature_map2)
30
31 # Tính độ tương đồng
32 similarity = compute_similarity(gram1, gram2)
33 print("Similarity Score:", similarity)
```

## Kết quả mong muốn

Sau khi chạy đoạn code trên, kết quả đầu ra sẽ là:

Similarity Score = 0.67

Điều này cho thấy hai ảnh có phong cách khá giống nhau với độ tương đồng 67%.

# Numpy - Term frequency and Invert Document Frequency

Hoàng-Nguyễn Vũ

## 1. Mô tả: Term Frequency-inverse document frequency

- **TF-IDF** (term frequency-inverse document frequency) là một kỹ thuật phổ biến trong lĩnh vực xử lý ngôn ngữ tự nhiên và khai thác văn bản. Mục đích của TF-IDF là để đánh giá tầm quan trọng của một từ trong một tài liệu so với toàn bộ tập hợp các tài liệu (corpus).

### Term Frequency (TF)

Do lường tần suất xuất hiện của một từ trong một tài liệu. Công thức tính:

$$TF(t, d) = \frac{\text{số lần xuất hiện của từ } t \text{ trong tài liệu } d}{\text{tổng số từ trong tài liệu } d}$$

### Inverse Document Frequency (IDF)

Do lường mức độ phổ biến của một từ trong tập hợp các tài liệu. Công thức tính:

$$IDF(t, D) = \log \left( \frac{\text{tổng số tài liệu } |D|}{1 + \text{số tài liệu chứa từ } t} \right)$$

Trong đó,  $|D|$  là tổng số tài liệu và số tài liệu chứa từ  $t$  được cộng thêm 1 để tránh chia cho 0.

### TF-IDF

Là tích của TF và IDF, giúp xác định tầm quan trọng của một từ trong một tài liệu cụ thể, đồng thời giảm thiểu ảnh hưởng của các từ phổ biến nhưng ít mang ý nghĩa.

$$TF-IDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

TF-IDF giúp phân biệt các từ quan trọng đối với nội dung của một tài liệu so với các từ xuất hiện thường xuyên trong nhiều tài liệu khác nhau nhưng ít mang giá trị thông tin.

2. **Bài tập:** Viết chương trình Python để tính toán giá trị TF-IDF của các từ trong một List gồm các câu: ["Tôi thích học AI", "AI là trí tuệ nhân tạo", "AGI là siêu trí tuệ nhân tạo"] và sử dụng thư viện numpy để hỗ trợ trong việc tính toán các ma trận. Biết các

```
1 import numpy as np
2 import math
3
4 # Bước 1: Tạo tập tài liệu mẫu
5 documents = ["Tôi thích học AI", "AI là trí tuệ nhân tạo", "AGI là siêu trí tuệ nhân tạo"]
6
7 # Bước 2: Tiền xử lý - tách từ và tính tần số
8 def compute_tf(doc):
9     ## Your code here ##
10
11 # Bước 3: Tính toán IDF
12 def compute_idf(docs):
13     ## Your code here ##
14
15 # Bước 4: Tính toán TF-IDF
16 def compute_tf_idf(tf, idf):
17     ## Your code here ##
18
19 # In kết quả
20 ## Your code here ##
```

Kết quả:

Tài liệu 1:

Tôi: 0.1014  
thích: 0.1014  
học: 0.1014  
AI: 0.0000

Tài liệu 2:

AI: 0.0000  
là: 0.0000  
trí: 0.0000  
tuệ: 0.0000  
nhân: 0.0000  
tạo: 0.0000

Tài liệu 3:

AGI: 0.0579  
là: 0.0000  
siêu: 0.0579  
trí: 0.0000  
tuệ: 0.0000  
nhân: 0.0000  
tạo: 0.0000

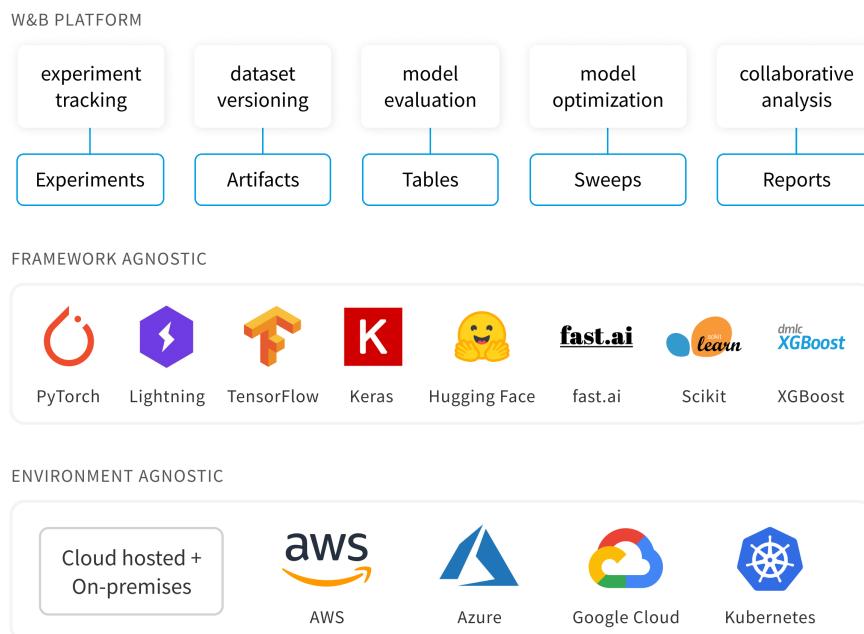
# Basic Python - Getting Started with WandB

*Hoàng-Nguyễn Vũ và Quang-Vinh Dinh*

# Weights & Biases

## 1. Mô tả:

- WandB hay còn gọi là **Weights and Biases** là một công cụ giúp các Data Scientist theo dõi mô hình, dữ liệu, thông tin hệ thống chỉ với vài dòng code. Weights & Bias hỗ trợ miễn phí cho các cá nhân hoặc tổ chức nghiên cứu. Công cụ này hỗ trợ rất nhiều nền tảng mà không cần chuyển đổi sang công cụ khác như TensorFlow, Keras, PyTorch, Sklearn, FastAI, ...
- Các thông tin cần quản lý và theo dõi sẽ được chuyển tới một giao diện (UI) của Weights & Biases. Giao diện này sẽ hiển thị và phân tích thông tin dễ dàng, việc so sánh các mô hình cũng như các tham số của nhiều thử nghiệm cũng diễn ra một cách nhanh chóng và trực tiếp. Một tiện ích nữa là bạn có thể chia sẻ các thông tin này cho team member của bạn. Dưới đây là một số công cụ trong WandB hỗ trợ bao gồm:



Hình 1: Tổng quan các công cụ của WandB.

- **Experiments:** Theo dõi các thử nghiệm trên các biểu đồ như evaluate function, loss function; thông tin về hệ thống như dung lượng bộ nhớ, thông tin xử lý GPU, ...
- **Artifacts:** Phiên bản dữ liệu, các phiên bản mô hình
- **Table:** Hiển thị các thông tin về các thử nghiệm và giá trị của các tham biến.
- **Sweeps:** Tối ưu hóa các tham số
- **Report:** Lưu trữ và chia sẻ các số liệu để có thể tái tạo mô hình

## 2. Cách triển khai WandB:

- **Tạo tài khoản:** Để bắt đầu, chúng ta cần tạo một tài khoản trên trang của Weights & Biases [tại đây](#). Sau khi đăng ký thành công, tại màn hình chính của trang web, chúng ta sẽ được cấp một API key để đăng nhập vào và sử dụng thư viện WandB.



### Quickstart: Tracking your first run in Weights & Biases

Weights & Biases' tools make it easy for you to quickly track experiments, visualize results, spot regressions, and more. Simply put, Weights & Biases enables you to build better models faster and easily share findings with colleagues.

Visualize your model training with [Python / Pytorch](#) or [Open in Colab](#)

#### 1. Set up the wandb library

Install the CLI and Python library for interacting with the Weights and Biases API.

```
pip install wandb
```

Next, log in and paste your API key when prompted.

```
wandb login
```



Your API key for logging in to the wandb library.

d0c

sd123...

Hình 2: Giao diện WandB sau khi đăng ký thành công.

#### • Cài đặt và đăng nhập vào thư viện WandB:

- Để cài đặt thư viện WandB, chúng ta sẽ thực thi dòng lệnh sau và đăng nhập dựa trên API Key được cấp ở bước đăng ký ở trên.

```
1 # Cài đặt thư viện WandB
2 !pip install wandb
3
4 # Đăng nhập với API Key
5 !wandb login
```

```
▶ 1 !wandb login
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter, or press ctrl+c to quit: .....  


```

Hình 3: Đăng nhập vào WandB với API Key.

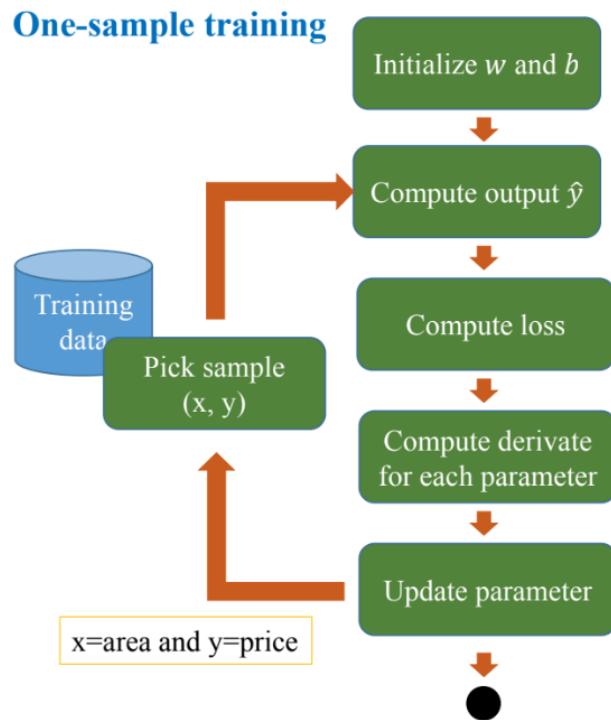
- **Làm quen với WandB:**

- Giả sử, chúng ta có tập dữ liệu giá nhà theo diện tích như bảng dưới, và chúng ta cần xây dựng mô hình hồi quy tuyến tính (Linear Regression) cơ bản để dự đoán giá nhà theo diện tích:

Bảng 1: Bảng dữ liệu giá nhà theo diện tích

Diện tích ( $m^2$ )	Giá
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7

- Để xây dựng mô hình linear regression với tập dữ liệu trên, chúng ta sẽ xây dựng dựa theo lưu đồ dưới đây:



Hình 4: Lưu đồ cài đặt mô hình Linear Regression

Chúng ta sẽ cài đặt và theo dõi quá trình training của mô hình với WandB như sau:

#### A. Import thư viện và khởi tạo dataset:

```

1 import pandas as pd
2 import wandb
3
4 areas = [6.7, 4.6, 3.5, 5.5]
5 prices = [9.1, 5.9, 4.6, 6.7]
6
7 dataset = pd.DataFrame({
8     'areas': areas,
9     'prices': prices
10})

```

#### B. Cài đặt mô hình Linear Regression kèm theo dõi quá trình huấn luyện với WandB:

- + Cài đặt mô hình:

```

1 # forward
2 def predict(x, w, b):
3     return x*w + b
4
5 # compute gradient
6 def gradient(y_hat, y, x):
7     dw = 2*x*(y_hat-y)
8     db = 2*(y_hat-y)
9
10    return (dw, db)
11
12 # update weights
13 def update_weight(w, b, lr, dw, db):
14     w_new = w - lr*dw
15     b_new = b - lr*db
16
17    return (w_new, b_new)

```

- + Huấn luyện mô hình và theo dõi quá trình với WandB:

```

1 # init weights
2 b = 0.04
3 w = -0.34
4 lr = 0.01
5 epochs = 10
6
7 # init project wandb
8 wandb.init(
9     # Set the project where this run will be logged
10     project="demo-linear-regression",
11     config={
12         "learning_rate": lr,
13         "epochs": epochs,
14     },
15 )

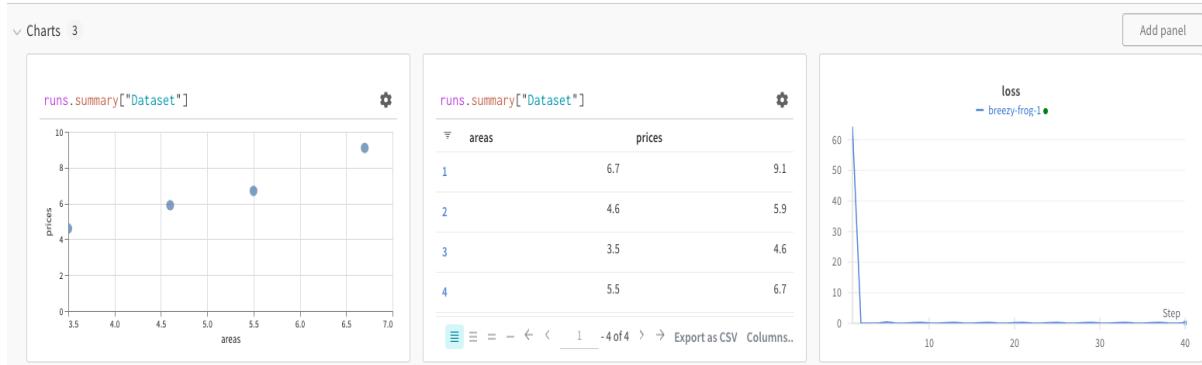
```

```

16
17 wandb.run.log({"Dataset" : wandb.Table(dataframe=dataset)})
18
19 X_train = dataset['areas']
20 Y_train = dataset['prices']
21
22 N = len(X_train)
23 # parameter
24 losses = [] # for debug
25
26 for epoch in range(epochs):
27     # for an epoch
28     for i in range(N):
29         # get a sample
30         x = X_train[i]
31         y = Y_train[i]
32
33         # predict y_hat
34         y_hat = predict(x, w, b)
35
36         # compute loss
37         loss = (y_hat-y)*(y_hat-y) / 2.0
38
39         # tracking loss with WandB
40         wandb.log({"loss": loss})
41
42         # compute gradient
43         (dw, db) = gradient(y_hat, y, x)
44
45         # update weights
46         (w, b) = update_weight(w, b, lr, dw, db)
47
48 # Mark a run as finished, and finish uploading all data.
49 wandb.finish()

```

- Trong quá trình training mô hình, chúng ta có thể theo dõi trực tiếp tại trang web của WandB, với kết quả training khi thực thi dòng lệnh trên như sau:

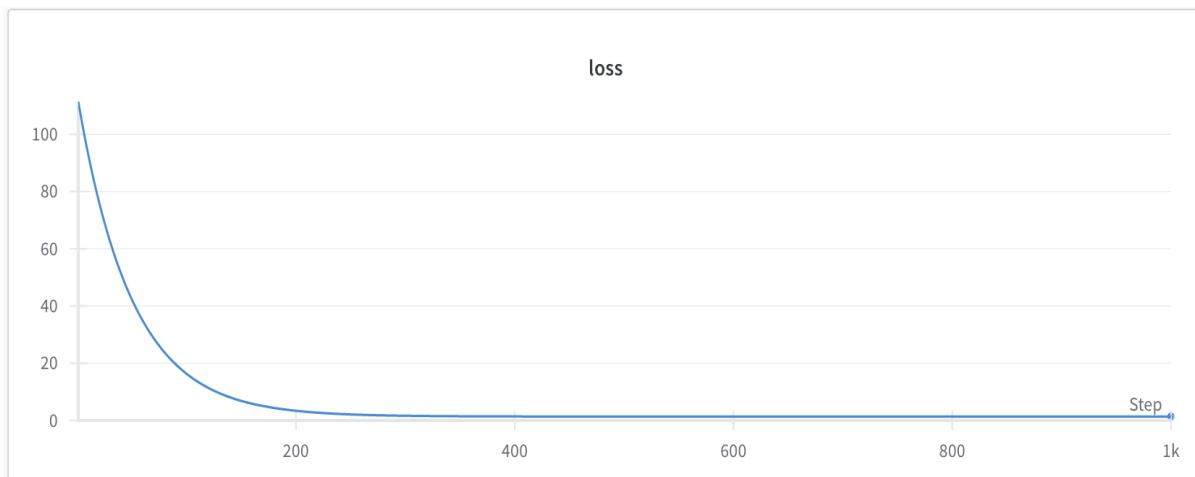


Hình 5: Kết quả theo dõi quá trình training mô hình Linear Regression

→ **Nhận xét:** Qua biểu đồ hàm loss được theo dõi ở trên, giúp chúng ta dễ dàng theo dõi quá trình training của mô hình.

3. **Bài tập:** Hãy đọc dữ liệu tại file sau: [advertising.csv](#). Xây dựng mô hình Linear Regression với 1000 epochs, khởi tạo các giá trị trọng số  $w_1 = w_2 = w_3 = 0$ ,  $b = 1$  và thực hiện tracking quá trình training với WandB cho tập data trên.

→ **Kết quả:**



Tables 1

	TV	Radio	Newspaper	Sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	12
4	151.5	41.3	58.5	16.5
5	180.8	10.8	58.4	17.9
6	8.7	48.9	75	7.2
7	57.5	32.8	23.5	11.8

Hình 6: Kết quả theo dõi quá trình training mô hình Linear Regression

- Hết -

AI VIET NAM – Warmup 2025

# Python OOP – Exercise

Ngày 6 tháng 6 năm 2025

## 1 Giới Thiệu Về Lập Trình Hướng Đối Tượng (OOP)

Lập trình hướng đối tượng (Object-Oriented Programming - OOP) là một phương pháp lập trình tổ chức code thành các đối tượng có thuộc tính (data) và phương thức (hành vi). OOP giúp cho việc tái sử dụng, mở rộng, và bảo trì code dễ dàng hơn. Python là một ngôn ngữ hỗ trợ OOP một cách linh hoạt và hiệu quả.

## 2 Bốn Nguyên Tắc Của OOP

- **Đóng gói (Encapsulation):** Giới hạn truy cập vào dữ liệu, chỉ cho phép truy cập thông qua các phương thức của class.
- **Kế thừa (Inheritance):** Cho phép một class sử dụng lại code từ class khác.
- **Đa hình (Polymorphism):** Cùng một phương thức nhưng hoạt động khác nhau dựa trên đối tượng sử dụng.
- **Trừu tượng (Abstraction):** Chỉ cung cấp những gì cần thiết và ẩn bớt những chi tiết không quan trọng.

## 3 Cấu Trúc Cơ Bản Của OOP Trong Python

### 3.1 Lớp (Class) và Đối Tượng (Object)

Lớp là khuôn mẫu chứa thuộc tính và phương thức. Đối tượng là một thể hiện của lớp.

```
1 class Animal:
2     def __init__(self, name):
3         self.name = name
4
5     def make_sound(self):
6         return "Some generic sound"
7
8 # Tạo đối tượng từ lớp Animal
9
10 dog = Animal("Buddy")
11 print(dog.name) # Output: Buddy
12 print(dog.make_sound()) # Output: Some generic sound
```

### 3.2 Đóng Gói (Encapsulation)

```

1 class BankAccount:
2     def __init__(self, owner, balance):
3         self.owner = owner
4         self.__balance = balance # Thuộc tính private
5
6     def deposit(self, amount):
7         self.__balance += amount
8         return f"Deposited {amount}, New balance: {self.__balance}"
9
10 Sử dụng class
11
12 account = BankAccount("Alice", 1000)
13 print(account.deposit(500)) # Output: Deposited 500, New balance: 1500

```

### 3.3 Kế Thừa (Inheritance)

```

1 class Dog(Animal): # Lớp Dog kế thừa từ Animal
2     def make_sound(self):
3         return "Woof! Woof!"
4
5 dog = Dog("Max")
6 print(dog.make_sound()) # Output: Woof! Woof!

```

### 3.4 Đa Hình (Polymorphism)

```

1 class Cat(Animal):
2     def make_sound(self):
3         return "Meow! Meow!"
4
5 animals = [Dog("Buddy"), Cat("Whiskers")]
6 for animal in animals:
7     print(f"{animal.name} says {animal.make_sound()}")

```

### 3.5 Trừu Tượng (Abstraction)

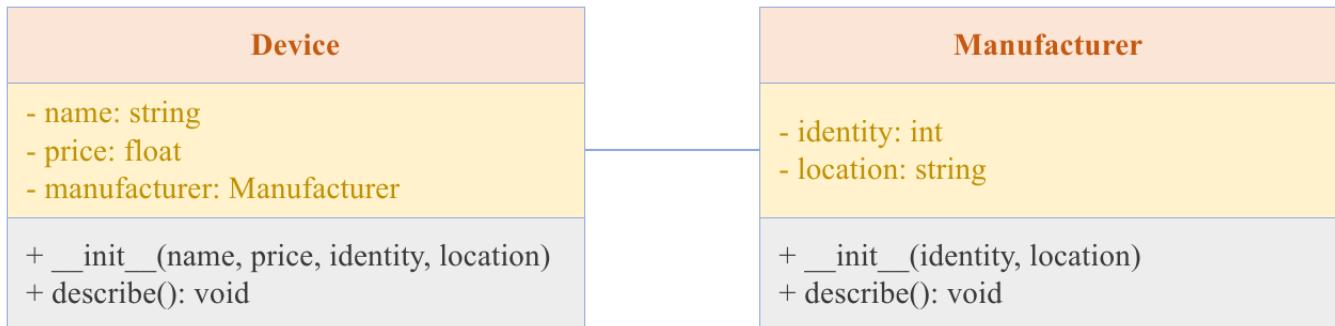
```

1 from abc import ABC, abstractmethod
2
3 class Vehicle(ABC):
4     @abstractmethod
5     def move(self):
6         pass
7
8 class Car(Vehicle):
9     def move(self):
10        return "Car is moving on the road"
11
12 class Boat(Vehicle):
13     def move(self):
14        return "Boat is sailing on the water"
15
16 car = Car()
17 boat = Boat()
18 print(car.move()) # Output: Car is moving on the road
19 print(boat.move()) # Output: Boat is sailing on the water

```

## 4 Bài Tập

- Thực hiện theo yêu cầu như class diagram bên dưới.



Hình 1: Class Diagram

```

1 # Examples 1
2 device1 = Device(name="mouse", price=2.5, identity=9725, location="Vietnam")
3 device1.describe()
4 >> Name: mouse - Price: 2.5
5 Identity: 9725 - Location: Vietnam
6
7 device2 = Device(name="monitor", price=12.5, identity=11, location="Germany")
8 device2.describe()
9 >> Name: monitor - Price: 12.5
10 Identity: 11 - Location: Germany
  
```

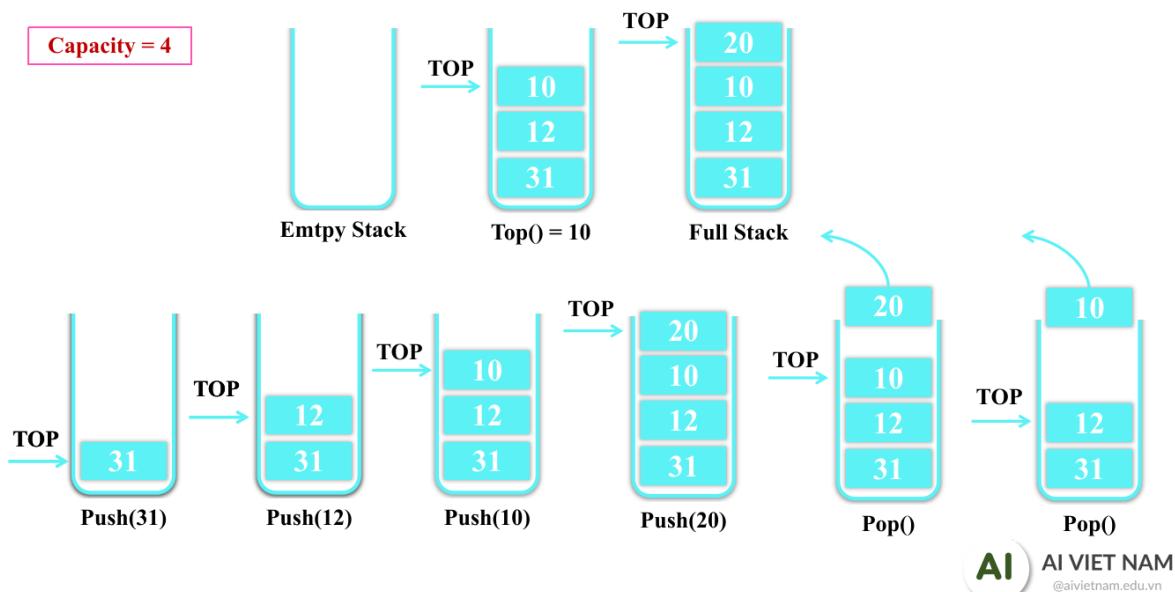
- Một Ward gồm có name (string) và danh sách của mọi người trong Ward. Một người person có thể là student, doctor, hoặc teacher. Một student gồm có name, yob (int) (năm sinh), và grade (string). Một teacher gồm có name, yob, và subject (string). Một doctor gồm có name, yob, và specialist (string). Lưu ý cần sử dụng a list để chứa danh sách của mọi người trong Ward.

- Thực hiện các class student, doctor, và teacher theo mô tả trên. Thực hiện describe() method để print ra tất cả thông tin của các objects.
- Viết addPerson(person) method trong Ward class để add thêm một người mới với nghề nghiệp bất kỳ (student, teacher, doctor) vào danh sách người của ward. Tạo ra một ward object, và thêm vào 1 student, 2 teacher, và 2 doctor. Thực hiện describe() method để in ra tên ward (name) và toàn bộ thông tin của từng người trong ward.
- Viết countDoctor() method để đếm số lượng doctor trong ward.
- Viết sortAge() method để sort mọi người trong ward theo tuổi của họ với thứ tự tăng dần. (hint: Có thể sử dụng sort của list hoặc viết thêm function đều được)
- Viết aveTeacherYearOfBirth() method để tính trung bình năm sinh của các teachers trong ward.

```

1 # Examples
2 # 2(a)
3 student1 = Student(name="studentA", yob=2010, grade="7")
4 student1.describe()
5 #output
6 >> Student - Name: studentA - YoB: 2010 - Grade: 7
  
```

```
7
8 teacher1 = Teacher(name="teacherA", yob=1969, subject="Math")
9 teacher1.describe()
10 #output
11 >> Teacher - Name: teacherA - YoB: 1969 - Subject: Math
12
13 doctor1 = Doctor(name="doctorA", yob=1945, specialist="Endocrinologists")
14 doctor1.describe()
15 #output
16 >> Doctor - Name: doctorA - YoB: 1945 - Specialist: Endocrinologists
17
18
19 # 2(b)
20 print()
21 teacher2 = Teacher(name="teacherB", yob=1995, subject="History")
22 doctor2 = Doctor(name="doctorB", yob=1975, specialist="Cardiologists")
23 ward1 = Ward(name="Ward1")
24 ward1.addPerson(student1)
25 ward1.addPerson(teacher1)
26 ward1.addPerson(teacher2)
27 ward1.addPerson(doctor1)
28 ward1.addPerson(doctor2)
29 ward1.describe()
30
31 #output
32 >> Ward Name: Ward1
33 Student - Name: studentA - YoB: 2010 - Grade: 7
34 Teacher - Name: teacherA - YoB: 1969 - Subject: Math
35 Teacher - Name: teacherB - YoB: 1995 - Subject: History
36 Doctor - Name: doctorA - YoB: 1945 - Specialist: Endocrinologists
37 Doctor - Name: doctorB - YoB: 1975 - Specialist: Cardiologists
38
39 # 2(c)
40 print(f"\nNumber of doctors: {ward1.countDoctor()}")
41
42 #output
43 >> Number of doctors: 2
44
45 # 2(d)
46 print("\nAfter sorting Age of Ward1 people")
47 ward1.sortAge()
48 ward1.describe()
49
50 #output
51 >> After sorting Age of Ward1 people
52 Ward Name: Ward1
53 Student - Name: studentA - YoB: 2010 - Grade: 7
54 Teacher - Name: teacherB - YoB: 1995 - Subject: History
55 Doctor - Name: doctorB - YoB: 1975 - Specialist: Cardiologists
56 Teacher - Name: teacherA - YoB: 1969 - Subject: Math
57 Doctor - Name: doctorA - YoB: 1945 - Specialist: Endocrinologists
58
59 # 2(e)
60 print(f"\nAverage year of birth (teachers): {ward1.aveTeacherYearOfBirth()}")
61
62 #output
63 >> Average year of birth (teachers): 1982.0
```



Hình 2: Stack

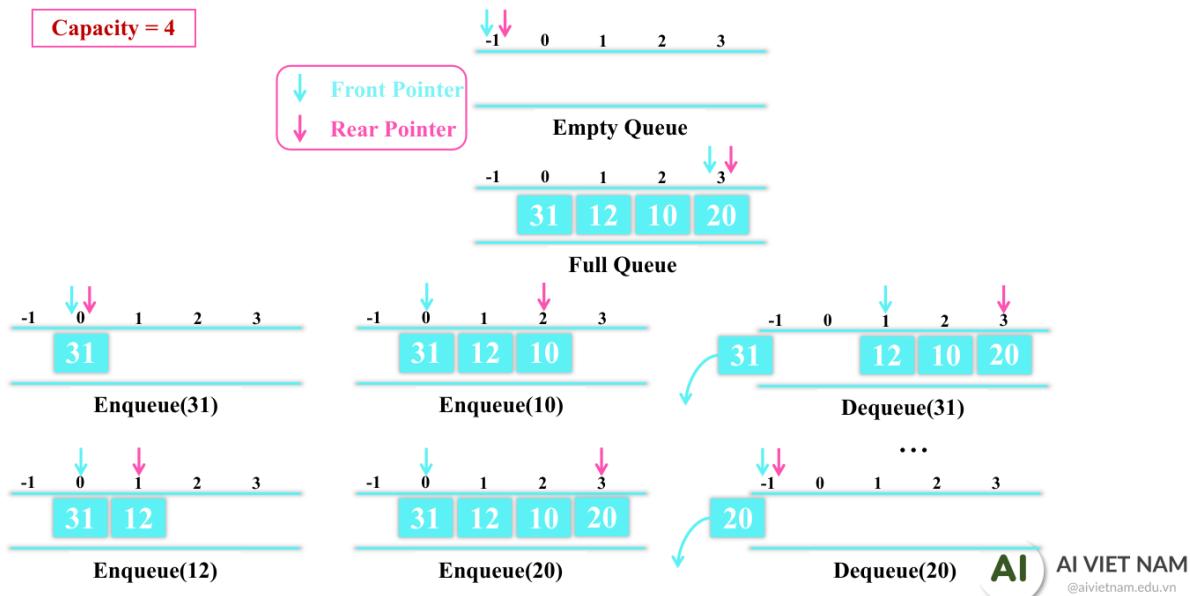
### 3. Thực hiện xây dựng class Stack với các chức năng (method) sau đây

- **initialization** method nhận một input "capacity": dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa
- **.isEmpty()**: kiểm tra stack có đang rỗng
- **.isFull()**: kiểm tra stack đã full chưa
- **.pop()**: loại bỏ top element và trả về giá trị đó
- **.push(value)** add thêm value vào trong stack
- **.top()** lấy giá trị top element hiện tại của stack, nhưng không loại bỏ giá trị đó
- **Không cần thiết phải thực hiện với pointer như trong hình minh họa**

```

1 stack1 = MyStack(capacity=5)
2
3 stack1.push(1)
4
5 stack1.push(2)
6
7 print(stack1.isEmpty())
8 >> False
9
10 print(stack1.top())
11 >> 2
12
13 print(stack1.pop())
14 >> 2
15
16 print(stack1.top())
17 >> 1
18
19 print(stack1.pop())
20 >> 1
21
22 print(stack1.isEmpty())
23 >> True

```



Hình 3: Queue

#### 4. Thực hiện xây dựng class Queue với các chức năng (method) sau đây

- **initialization** method nhận một input "capacity": dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa
- **.isEmpty()**: kiểm tra queue có đang rỗng
- **.isFull()**: kiểm tra queue đã full chưa
- **.dequeue()**: loại bỏ first element và trả về giá trị đó
- **.enqueue(value)** add thêm value vào trong queue
- **.front()** lấy giá trị first element hiện tại của queue, nhưng không loại bỏ giá trị đó
- **Không cần thiết phải thực hiện với các pointers như trong hình minh họa**

```

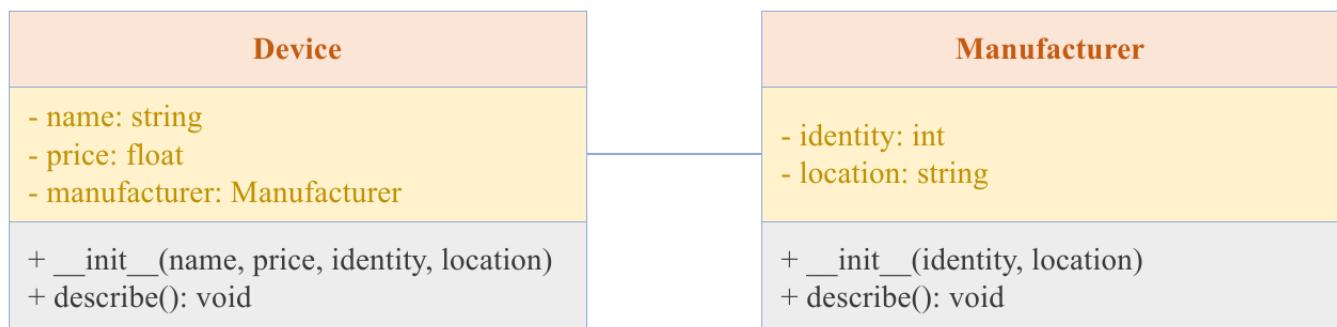
1 queue1 = MyQueue(capacity=5)
2
3 queue1.enqueue(1)
4
5 queue1.enqueue(2)
6
7 print(queue1.isFull())
8 >> False
9
10 print(queue1.front())
11 >> 1
12
13 print(queue1.dequeue())
14 >> 1
15
16 print(queue1.front())
17 >> 2
18
19 print(queue1.dequeue())
20 >> 2
21
22 print(queue1.isEmpty())
23 >> True

```

## II. Câu hỏi trắc nghiệm

- Đọc tự luận trước để nắm được idea tổng quát (sẽ không yêu cầu nhưng khuyến khích các bạn tự làm tự luận) và các bài này sẽ được giải trong buổi TA.
- Các bạn phải làm phần trắc nghiệm
  - Các câu hỏi có ký hiệu (**LT**): là các câu hỏi lý thuyết và/hoặc đã có sẵn trong file hint nên các bạn chỉ cần hiểu và chạy lại để chọn được đáp án đúng
  - Các câu hỏi có ký hiệu (**Code**): là câu hỏi yêu cầu các bạn phải trực tiếp code vào phần bị khuyết để có thể chọn được đáp án đúng
  - Lưu ý:** Đối với dạng câu hỏi (**Code**) trong file hint luôn có 1 test case bắt đầu với từ khóa assert nếu các bạn chạy không báo lỗi có nghĩa các bạn đã vượt qua được test case này và chạy lệnh tiếp theo để trả lời câu hỏi trắc nghiệm
  - Lưu ý:** Đọc kỹ các code gợi ý và code ví dụ mẫu ở tự luận có thể sẽ có ích cho các bạn khi làm trắc nghiệm

**Câu hỏi 1 (LT)** : Theo thiết kế này thì mối quan hệ giữa 2 class này được gọi là gì?



Hình 4: Class Diagram

- Aggregation
- Composition
- Inheritance
- Tất cả đều sai

**Câu hỏi 2 (LT)** : Đầu ra của chương trình sau đây là gì?

```

1 class Manufacturer:
2     def __init__(self, identity:int, location: str ):
3         self.__identity = identity
4         self.__location = location
5
6     def describe(self):
7         print(f"Identity: {self.__identity} - Location: {self.__location}")
8
9 manu1 = Manufacturer(identity=100, location='Vietnam')
10 manu1.describe()

```

- a) Identity: 100 - Location: Vietnam
- b) Identity: Vietnam - Location: 100
- c) None
- d) Raise an error

**Câu hỏi 3 (LT)** : Đoạn code sau đây thể hiện mối quan hệ gì?

```

1 class Manufacturer:
2     def __init__(self, identity:int, location: str ):
3         self.__identity = identity
4         self.__location = location
5
6     def describe(self):
7         print(f"Identity: {self.__identity} - Location: {self.__location}")
8
9
10 class Device:
11     def __init__(self, name:str, price:float, identity:int, location:str ):
12         self.__name = name
13         self.__price = price
14         self.__manufacturer = Manufacturer(identity, location)
15
16     def describe(self):
17         print(f"Name: {self.__name} - Price: {self.__price}")
18         self.__manufacturer.describe()
19 device1 = Device(name="touchpad", price=3.3, identity=1111, location="Vietnam")
20 device1.describe()

```

- a) Aggregation
- b) Composition
- c) Inheritance
- d) Tất cả đều sai

**Câu hỏi 4 (LT)** : Đoạn code sau đây thể hiện mối quan hệ gì?

```

1 class Manufacturer:
2     def __init__(self, identity:int, location: str ):
3         self.__identity = identity
4         self.__location = location
5
6     def describe(self):
7         print(f"Identity: {self.__identity} - Location: {self.__location}")
8
9
10 class Device:
11     def __init__(self, name:str, price:float, manu:Manufacturer):
12         self.__name = name
13         self.__price = price
14         self.__manufacturer = manu
15
16     def describe(self):
17         print(f"Name: {self.__name} - Price: {self.__price}")
18         self.__manufacturer.describe()
19
20 manu1 = Manufacturer(identity=1111, location='Vietnam')
21 device1 = Device(name="touchpad", price=3.3, manu=manu1)
22 device1.describe()

```

- a) Aggregation

- b) Composition
- c) Inheritance
- d) Tất cả đều sai

**Câu hỏi 5 (Code)** : Một người (person) có thể là student, doctor, hoặc teacher. Một student gồm có name (string), yob (int) (năm sinh), và grade (string). Các bạn thực hiện viết class Student theo mô tả trên (Các bạn sẽ viết thêm describe() method để print ra tất cả thông tin của object) và kết quả đầu ra là gì? Chọn đáp án đúng nhất bên dưới.

```

1 from abc import ABC, abstractmethod
2
3 class Person(ABC):
4     def __init__(self, name:str, yob:int):
5         self._name = name
6         self._yob = yob
7
8     def getYoB(self):
9         return self._yob
10
11 @abstractmethod
12 def describe(self):
13     pass
14
15
16 class Student(Person):
17     def __init__(self, name:str, yob:int, grade:str):
18         ##### YOUR CODE HERE #####
19
20         #####
21     def describe(self):
22         ##### YOUR CODE HERE #####
23
24         #####
25
26 student1 = Student(name="studentZ2023", yob=2011, grade="6")
27 student1.describe()

```

- a) Student - Name: studentZ2023 - YoB: 2011 - Grade: 6
- b) Student - Name: studentZ2023 - YoB: 6 - Grade: 2011
- c) Student - Name: 6 - YoB: studentZ2023 - Grade: 2011
- d) Tất cả đều sai

**Câu hỏi 6 (Code)** : Một người (person) có thể là student, doctor, hoặc teacher. Một teacher gồm có name (string), yob (int), và subject (string). Các bạn thực hiện viết class Teacher theo mô tả trên (Các bạn sẽ viết thêm describe() method để print ra tất cả thông tin của object) và kết quả đầu ra là gì? Chọn đáp án đúng nhất bên dưới.

```

1 from abc import ABC, abstractmethod
2
3 class Person(ABC):
4     def __init__(self, name:str, yob:int):
5         self._name = name
6         self._yob = yob
7
8     def getYoB(self):
9         return self._yob
10
11 @abstractmethod

```

```

12     def describe(self):
13         pass
14
15
16 class Teacher(Person):
17     def __init__(self, name:str, yob:int, subject:str):
18         ##### YOUR CODE HERE#####
19
20         #####
21
22     def describe(self):
23         ##### YOUR CODE HERE#####
24
25         #####
26
27 teacher1 = Teacher(name="teacherZ2023", yob=1991, subject="History")
28 teacher1.describe()

```

- a) Teacher - Name: teacherZ2023 - YoB: 1991 - Subject: History
- b) Teacher - Name: 1991 - YoB: teacherZ2023 - Subject: History
- c) Teacher - Name: History - YoB: teacherZ2023 - Subject: 1991
- d) Tất cả đều sai

**Câu hỏi 7 (Code)** : Một người (person) có thể là student, doctor, hoặc teacher. Một doctor gồm có name (string), yob (string), và specialist (string). Các bạn thực hiện viết class Teacher theo mô tả trên (Các bạn sẽ viết thêm describe() method để print ra tất cả thông tin của object) và kết quả đầu ra là gì? Chọn đáp án đúng nhất bên dưới.

```

1 from abc import ABC, abstractmethod
2
3 class Person(ABC):
4     def __init__(self, name:str, yob:int):
5         self._name = name
6         self._yob = yob
7
8     def getYoB(self):
9         return self._yob
10
11     @abstractmethod
12     def describe(self):
13         pass
14
15
16 class Doctor(Person):
17     def __init__(self, name:str, yob:int, specialist:str):
18         ##### YOUR CODE HERE#####
19
20         #####
21
22     def describe(self):
23         ##### YOUR CODE HERE#####
24
25         #####
26
27 doctor1 = Doctor(name="doctorZ2023", yob=1981, specialist="Endocrinologists")
28 doctor1.describe()

```

- a) Doctor - Name: doctorZ2023 - YoB: 1981 - Specialist: Endocrinologists
- b) Doctor - Name: 1981 - YoB: doctorZ2023 - Specialist: Endocrinologists

- c) Doctor - Name: Endocrinologists - YoB: 1981 - Specialist: doctorZ2023  
d) Tất cả đều sai

**Câu hỏi 8 (Code)** : Một Ward gồm có name (string) và danh sách của mọi người trong Ward. Một người person có thể là student, doctor, hoặc teacher và cần sử dụng một list để chứa danh sách của mọi người trong Ward. Viết addPerson(person) method trong Ward class để add thêm một người mới với nghề nghiệp bất kỳ (student, teacher, doctor) vào danh sách người của ward. Tạo ra một ward object, và thêm vào 1 student, 2 teacher, và 2 doctor (sử dụng code ở câu 5,6,7 để tạo person mong muốn). Thực hiện describe() method để in ra tên ward (name) và toàn bộ thông tin của từng người trong ward. Chọn đáp án đúng nhất bên dưới.

```

1 class Ward:
2     def __init__(self, name:str):
3         ##### YOUR CODE HERE#####
4
5         #####
6
7     def addPerson(self, person:Person):
8         ##### YOUR CODE HERE#####
9
10    #####
11
12    def describe(self):
13        ##### YOUR CODE HERE#####
14
15        #####
16
17 student1 = Student(name="studentK-111", yob=2012, grade="5")
18 teacher1 = Teacher(name="teacherK-222", yob=1966, subject="Math")
19 doctor1 = Doctor(name="doctorK-333", yob=1965, specialist="Endocrinologists")
20 teacher2 = Teacher(name="teacherK-444", yob=1945, subject="History")
21 doctor2 = Doctor(name="doctorK-555", yob=1975, specialist="Cardiologists")
22 ward1 = Ward(name="Ward11")
23 ward1.addPerson(student1)
24 ward1.addPerson(teacher1)
25 ward1.addPerson(teacher2)
26 ward1.addPerson(doctor1)
27 ward1.addPerson(doctor2)
28 ward1.describe()

```

- a) Ward Name: Ward11  
Student - Name: studentK-111 - YoB: 2012 - Grade: 5  
Teacher - Name: teacherK-222 - YoB: 1966 - Subject: Math  
Teacher - Name: teacherK-444 - YoB: 1945 - Subject: History  
Doctor - Name: doctorK-333 - YoB: 1965 - Specialist: Endocrinologists  
Doctor - Name: doctorK-555 - YoB: 1975 - Specialist: Cardiologists  
b) Ward Name: Ward22  
Student - Name: studentK-111 - YoB: 2012 - Grade: 5  
Teacher - Name: Math - YoB: teacherK-222 - Subject: 1966  
Teacher - Name: teacherK-444 - YoB: 1945 - Subject: History  
Doctor - Name: 1965 - YoB: doctorK-333 - Specialist: Endocrinologists  
Doctor - Name: doctorK-555 - YoB: 1975 - Specialist: Cardiologists  
c) Tất cả đều đúng  
d) Tất cả đều sai

**Câu hỏi 9 (LT)** : Thực hiện xây dựng class Stack với các chức năng (method) sau đây: **initialization**

method nhận một input "capacity": dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa. `.isEmpty()`: kiểm tra stack có đang rỗng. Kết quả đầu ra là gì?

```

1 class MyStack:
2     def __init__(self, capacity):
3         self.__capacity = capacity
4         self.__stack = []
5
6     def isEmpty(self):
7         return len(self.__stack) == 0
8
9 stack1 = MyStack(capacity=5)
10 print(stack1.isEmpty())

```

- a) True
- b) False
- c) None
- d) Raise an error

**Câu hỏi 10 (LT)** : Thực hiện xây dựng class Stack với các chức năng (method) sau đây: **initialization** method nhận một input "capacity": dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa (tại đây stack cũng đã được thêm vào các element). `.isFull()`: kiểm tra stack đã full chưa. Kết quả đầu ra là gì?

```

1 class MyStack:
2     def __init__(self, capacity):
3         self.__capacity = capacity
4         self.__stack = [1,2,3,4,5]
5
6     def isFull(self):
7         return len(self.__stack) == self.__capacity
8
9 stack1 = MyStack(capacity=5)
10 print(stack1.isFull())

```

- a) True
- b) False
- c) None
- d) Raise an error

**Câu hỏi 11 (Code)** : Thực hiện xây dựng class Stack với các chức năng (method) sau đây: **initialization** method nhận một input "capacity": dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa. `.isFull()`: kiểm tra stack đã full chưa. `.push(value)` add thêm value vào trong stack. Kết quả đầu ra là gì?

```

1 class MyStack:
2     def __init__(self, capacity):
3         self.__capacity = capacity
4         self.__stack = []
5
6     def isFull(self):
7         return len(self.__stack) == self.__capacity
8
9     def push(self, value):
10        ##### YOUR CODE HERE#####
11
12        #####

```

```

14 stack1 = MyStack(capacity=5)
15 stack1.push(1)
16 stack1.push(2)
17 print(stack1.isFull())

```

- a) True
- b) False
- c) None
- d) Raise an error

**Câu hỏi 12 (Code)** : Thực hiện xây dựng class Stack với các chức năng (method) sau đây: **initialization** method nhận một input "capacity": dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa. **.isEmpty()**: kiểm tra stack có đang rỗng. **.isFull()**: kiểm tra stack đã full chưa. **.push(value)** add thêm value vào trong stack. **.top()** lấy giá trị top element hiện tại của stack, nhưng không loại bỏ giá trị đó. Kết quả đầu ra là gì?

```

1 class MyStack:
2     def __init__(self, capacity):
3         self.__capacity = capacity
4         self.__stack = []
5
6     def isEmpty(self):
7         return len(self.__stack) == 0
8
9     def isFull(self):
10        return len(self.__stack) == self.__capacity
11
12    def push(self, value):
13        ##### YOUR CODE HERE#####
14
15        #####
16
17    def top(self):
18        ##### YOUR CODE HERE#####
19
20        #####
21
22 stack1 = MyStack(capacity=5)
23 stack1.push(1)
24 stack1.push(2)
25 print(stack1.top())

```

- a) 1
- b) 2
- c) None
- d) Raise an error

**Câu hỏi 13 (LT)** : Thực hiện xây dựng class Queue với các chức năng (method) sau đây **initialization** method nhận một input "capacity": dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa..**.isEmpty()**: kiểm tra queue có đang rỗng. Kết quả đầu ra là gì?

```

1 class MyQueue:
2     def __init__(self, capacity):
3         self.__capacity = capacity
4         self.__queue = []
5
6     def isEmpty(self):
7         return len(self.__queue) == 0

```

```

8
9 queue1 = MyQueue(capacity=5)
10 print(queue1.isEmpty())

```

- a) True
- b) False
- c) None
- d) Raise an error

**Câu hỏi 14 (LT)** : Thực hiện xây dựng class Queue với các chức năng (method) sau đây **initialization** method nhận một input "capacity": dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa. **.isFull()**: kiểm tra queue đã full chưa. Kết quả đầu ra là gì?

```

1 class MyQueue:
2     def __init__(self, capacity):
3         self.__capacity = capacity
4         self.__queue = []
5
6     def isFull(self):
7         return len(self.__queue) == self.__capacity
8
9
10 queue1 = MyQueue(capacity=5)
11 print(queue1.isFull())

```

- a) True
- b) False
- c) None
- d) Raise an error

**Câu hỏi 15 (Code)** : Thực hiện xây dựng class Queue với các chức năng (method) sau đây **initialization** method nhận một input "capacity": dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa. **.isFull()**: kiểm tra queue đã full chưa. **.enqueue(value)** add thêm value vào trong queue. Kết quả đầu ra là gì?

```

1 class MyQueue:
2     def __init__(self, capacity):
3         self.__capacity = capacity
4         self.__queue = []
5
6     def isFull(self):
7         return len(self.__queue) == self.__capacity
8
9     def enqueue(self, value):
10        ##### YOUR CODE HERE#####
11
12        #####
13
14 queue1 = MyQueue(capacity=5)
15 queue1.enqueue(1)
16 queue1.enqueue(2)
17 print(queue1.isFull())

```

- a) True
- b) False
- c) None
- d) Raise an error

**Câu hỏi 16 (Code)** : Thực hiện xây dựng class Queue với các chức năng (method) sau đây **initialization** method nhận một input "capacity": dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa. **.isFull()**: kiểm tra queue đã full chưa. **.enqueue(value)** add thêm value vào trong queue. **.front()** lấy giá trị first element hiện tại của queue, nhưng không loại bỏ giá trị đó . Kết quả đầu ra là gì?

```

1  class MyQueue:
2      def __init__(self, capacity):
3          self.__capacity = capacity
4          self.__queue = []
5
6      def isEmpty(self):
7          return len(self.__queue) == 0
8
9      def isFull(self):
10         return len(self.__queue) == self.__capacity
11
12     def enqueue(self, value):
13         ##### YOUR CODE HERE#####
14
15         #####
16
17     def front(self):
18         ##### YOUR CODE HERE#####
19
20         #####
21 queue1 = MyQueue(capacity=5)
22 queue1.enqueue(1)
23 queue1.enqueue(2)
24 print(queue1.front())

```

- a) 1
- b) 2
- c) None
- d) Raise an error

## Đáp án

Câu 1: D

Câu 2: B

Câu 3: B

Câu 4: A

Câu 5: A

Câu 6: A

Câu 7: A

Câu 8: A

Câu 9: B

Câu 10: A

Câu 11: B

Câu 12: A

Câu 13: A

Câu 14: B

Câu 15: B

Câu 16: A

# Linear Regression

*Hoàng-Nguyễn Vũ*

## 1 Lý thuyết về Linear Regression

### 1.1 Giới thiệu về Linear Regression

Hồi quy tuyến tính (**Linear Regression**) là một thuật toán trong Machine Learning được sử dụng để dự đoán một biến phụ thuộc ( $y$ ) dựa trên một hoặc nhiều biến độc lập ( $X$ ). Đây là một trong những mô hình đơn giản nhưng mạnh mẽ để phân tích dữ liệu.

Có hai loại chính:

- **Hồi quy tuyến tính đơn giản** (*Simple Linear Regression*) – chỉ có một biến đầu vào.
- **Hồi quy tuyến tính bội** (*Multiple Linear Regression*) – có nhiều biến đầu vào.

### 1.2 Công thức toán học

#### 1.2.1 Hồi quy tuyến tính đơn giản

Công thức tổng quát của hồi quy tuyến tính đơn giản:

$$y = wX + b$$

Trong đó:

- $y$  là giá trị đầu ra cần dự đoán.
- $X$  là biến đầu vào.
- $w$  là hệ số hồi quy.
- $b$  là hằng số (bias).

Ví dụ: Dự đoán giá nhà dựa trên diện tích:

$$\text{Giá nhà} = w \times \text{Diện tích} + b$$

#### 1.2.2 Hồi quy tuyến tính bội

Khi có nhiều biến đầu vào, công thức tổng quát là:

$$y = w_1X_1 + w_2X_2 + \cdots + w_nX_n + b$$

Hoặc viết dưới dạng ma trận:

$$Y = XW + b$$

### 1.3 Tìm tham số $w$ và $b$

Để tìm giá trị tối ưu của  $w$  và  $b$ , ta sử dụng phương pháp **Gradient Descent** để giảm sai số.

#### 1.3.1 Hàm mất mát (Loss Function)

Hàm mất mát phổ biến nhất là **Mean Squared Error (MSE)**:

$$MSE = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$$

#### 1.3.2 Gradient Descent

Gradient Descent là thuật toán tối ưu hóa giúp cập nhật  $w$  và  $b$  sao cho giảm MSE:

$$w = w - \alpha \cdot \frac{1}{m} \sum (y_i - \hat{y}_i) \cdot X_i$$

$$b = b - \alpha \cdot \frac{1}{m} \sum (y_i - \hat{y}_i)$$

### 1.4 Đánh giá mô hình

Sau khi huấn luyện mô hình hồi quy tuyến tính, ta cần đánh giá hiệu suất của nó để đảm bảo độ chính xác khi áp dụng vào thực tế. Có nhiều chỉ số được sử dụng để đánh giá một mô hình hồi quy tuyến tính, bao gồm:

- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE)
- R-squared Score ( $R^2$ )

Dưới đây là chi tiết về từng phương pháp đánh giá.

#### 1.4.1 Mean Squared Error (MSE)

MSE đo lường độ lệch trung bình bình phương giữa giá trị dự đoán và giá trị thực tế. Công thức tính MSE như sau:

$$MSE = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$$

Trong đó:

- $m$  là số lượng mẫu trong tập dữ liệu.

- $y_i$  là giá trị thực tế của mẫu  $i$ .
- $\hat{y}_i$  là giá trị dự đoán của mẫu  $i$ .

**Ý nghĩa:**

- MSE càng nhỏ chứng tỏ mô hình dự đoán càng chính xác.
- Do lỗi được bình phương, MSE nhạy cảm với outliers (các giá trị ngoại lai).

#### 1.4.2 Root Mean Squared Error (RMSE)

RMSE là căn bậc hai của MSE, giúp biểu diễn sai số theo cùng đơn vị với dữ liệu gốc:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2}$$

**Ý nghĩa:**

- RMSE có cùng đơn vị với đầu ra  $y$ , dễ diễn giải hơn MSE.
- Giá trị RMSE thấp chứng tỏ mô hình có độ chính xác cao.

#### 1.4.3 Mean Absolute Error (MAE)

MAE đo lường trung bình sai số tuyệt đối giữa giá trị thực tế và giá trị dự đoán:

$$MAE = \frac{1}{m} \sum_{i=1}^m |y_i - \hat{y}_i|$$

**Ý nghĩa:**

- MAE ít bị ảnh hưởng bởi outliers hơn MSE do không có bình phương sai số.
- Giá trị MAE càng nhỏ, mô hình càng tốt.

#### 1.4.4 R-squared Score ( $R^2$ )

$R^2$  (coefficient of determination) đo lường mức độ mô hình có thể giải thích được phuơng sai của dữ liệu. Công thức tính:

$$R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}$$

Trong đó:

- $\bar{y}$  là giá trị trung bình của  $y$ .
- $\sum (y_i - \hat{y}_i)^2$  là tổng phuơng sai của sai số dự đoán.

- $\sum(y_i - \bar{y})^2$  là tổng phuơng sai của dữ liệu thực tế.

**Ý nghĩa:**

- $R^2$  nằm trong khoảng  $[0, 1]$ . Giá trị càng cao chứng tỏ mô hình giải thích tốt dữ liệu.
- $R^2 = 1$  khi mô hình dự đoán hoàn hảo.
- $R^2 = 0$  khi mô hình không dự đoán tốt hơn một đường trung bình ngang.
- $R^2 < 0$  khi mô hình hoạt động tệ hơn cả việc đoán giá trị trung bình của dữ liệu.

#### 1.4.5 So sánh các phuơng pháp đánh giá

Chỉ số	Công thức	Đặc điểm
MSE	$\frac{1}{m} \sum (y_i - \hat{y}_i)^2$	Nhạy cảm với outliers
RMSE	$\sqrt{MSE}$	Dễ diễn giải, nhạy cảm với outliers
MAE	$\frac{1}{m} \sum  y_i - \hat{y}_i $	Ít bị ảnh hưởng bởi outliers
$R^2$ Score	$1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$	Đánh giá mức độ giải thích của mô hình

Bảng 1: So sánh các phuơng pháp đánh giá hồi quy tuyến tính

## 2 Bài tập

### Cài đặt hồi quy tuyến tính

Hãy đọc dữ liệu tại file sau: [advertising.csv](#). Xây dựng lớp `LinearRegression` để thực hiện hồi quy tuyến tính từ đầu bằng **Gradient Descent**.

#### Yêu cầu

1. Viết một class `LinearRegression` với các phuơng thức:
  - `fit(X, y)`: Huấn luyện mô hình bằng Gradient Descent.
  - `predict(X)`: Dự đoán giá trị  $y$  từ  $X$ .
  - `evaluate(X, y)`: Đánh giá mô hình bằng MSE.
  - `plot_regression_line(X, y)`: Vẽ đường hồi quy.
2. Tạo một bộ dữ liệu giả lập.
3. Huấn luyện mô hình và vẽ đường hồi quy.

## Gợi ý cài đặt (Python)

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 class LinearRegression:
5     def __init__(self, learning_rate=0.01, epochs=1000):
6         self.learning_rate = learning_rate
7         self.epochs = epochs
8         self.w = None
9         self.b = None
10
11    def fit(self, X, y):
12        m, n = X.shape
13        self.w = np.zeros(n)
14        self.b = 0
15
16        for _ in range(self.epochs):
17            ## Your code here ##
18
19    def predict(self, X):
20        ## Your code here ##
21
22    def evaluate(self, X, y):
23        ## Your code here ##
24
25    def plot_regression_line(self, X, y):
26        plt.scatter(X, y, color="blue", label="Data points")
27        y_pred = self.predict(X)
28        plt.plot(X, y_pred, color="red", label="Regression line")
29        plt.xlabel("X")
30        plt.ylabel("y")
31        plt.legend()
32        plt.show()
33
34 # Load data chia tỉ lệ: Train 80% và Test 20%
35 ## Your code here ###
36
37 # Huấn luyện mô hình
38 model = LinearRegression(learning_rate=0.01, epochs=1000)
39 model.fit(X_train, y_train)
40
41 # Dự đoán và vẽ đường hồi quy
42 print(f"MSE: {model.evaluate(X_test, y_test)}")
43 model.plot_regression_line(X_test, y_test)
```

# Tree data structure

Hoàng-Nguyễn Vũ

## 1 Lý thuyết về cấu trúc dữ liệu cây

### 1.1 Cấu trúc dữ liệu cây là gì?

Cấu trúc dữ liệu **cây** (Tree) là một tập hợp các nút (*nodes*) có mối quan hệ cha - con (*parent-child*). Đây là một dạng dữ liệu phi tuyến tính, được sử dụng để biểu diễn quan hệ phân cấp giữa các phần tử.

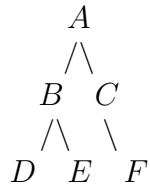
Một số ứng dụng thực tế của cây:

- **Hệ thống tệp tin** trong máy tính (cây thư mục).
- **Cây quyết định (Decision Tree)** trong học máy.
- **Cấu trúc tổ chức doanh nghiệp** (sơ đồ tổ chức).
- **Cây biểu thức (Expression Tree)** trong xử lý toán học.
- **Tìm kiếm và sắp xếp** (cây nhị phân tìm kiếm, cây AVL, cây đỏ-đen).

### 1.2 Các thành phần chính của cây

- **Nút (Node)**: Phần tử cơ bản của cây.
- **Nút gốc (Root)**: Nút đầu tiên trong cây, không có nút cha.
- **Nút con (Child)**: Các nút có liên kết trực tiếp từ một nút cha.
- **Nút cha (Parent)**: Một nút có các nút con.
- **Nút lá (Leaf)**: Nút không có nút con nào.
- **Cấp (Level)**: Độ sâu của một nút tính từ gốc.
- **Chiều cao của cây (Height)**: Số mức tối đa từ gốc đến nút xa nhất.

Ví dụ về một cây đơn giản:



## 2 Các ví dụ về cấu trúc cây và code mẫu

### 2.1 Cài đặt lớp TreeNode

Lớp TreeNode đại diện cho một nút trong cây:

```

1 class TreeNode:
2     def __init__(self, value):
3         self.value = value
4         self.children = []
5
6     def add_child(self, child):
7         self.children.append(child)
8
9     def remove_child(self, child):
10        self.children = [c for c in self.children if c != child]
11
12    def print_tree(self, level=0):
13        print(" " * (level * 4) + "|-- " + self.value)
14        for child in self.children:
15            child.print_tree(level + 1)

```

### 2.2 Cài đặt lớp Tree

Lớp Tree giúp quản lý toàn bộ cây:

```

16 class Tree:
17     def __init__(self, root_value):
18         self.root = TreeNode(root_value)
19
20     def find(self, value, node=None):
21         if node is None:
22             node = self.root
23         if node.value == value:
24             return node
25         for child in node.children:
26             found = self.find(value, child)
27             if found:
28                 return found
29         return None
30
31     def insert(self, parent_value, child_value):
32         parent_node = self.find(parent_value)
33         if parent_node:
34             parent_node.add_child(TreeNode(child_value))
35         else:
36             print(f"Nút cha '{parent_value}' không tồn tại.")
37
38     def print_tree(self):
39         self.root.print_tree()

```

### 3 Duyệt cây theo BFS (Breadth-First Search)

#### 3.1 BFS là gì?

BFS (Breadth-First Search) hay **Tìm kiếm theo chiều rộng** là một thuật toán duyệt cây bằng cách kiểm tra tất cả các nút ở mức hiện tại trước khi chuyển sang mức tiếp theo.

Ví dụ duyệt BFS trên cây:

$$A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F$$

#### 3.2 Cài đặt thuật toán BFS

```

1 from collections import deque
2
3 class Tree:
4     def __init__(self, root_value):
5         self.root = TreeNode(root_value)
6
7     def traverse_bfs(self):
8         if not self.root:
9             return
10
11         queue = deque([self.root])
12         while queue:
13             node = queue.popleft()
14             print(node.value, end=" ")
15             for child in node.children:
16                 queue.append(child)
17         print()
```

#### 3.3 Ví dụ sử dụng BFS

```

1 # Khởi tạo cây
2 tree = Tree("A")
3
4 # Thêm các nút con vào cây
5 tree.root.add_child(TreeNode("B"))
6 tree.root.add_child(TreeNode("C"))
7
8 tree.root.children[0].add_child(TreeNode("D"))
9 tree.root.children[0].add_child(TreeNode("E"))
10 tree.root.children[1].add_child(TreeNode("F"))
11
12 # In cây theo dạng phân cấp
13 print("Cây ban đầu:")
14 tree.root.print_tree()
15
16 # Duyệt cây theo BFS
17 print("\nDuyệt cây theo BFS:")
18 tree.traverse_bfs()
```

## 4 Bài tập

### 4.1 Bài 1: Cài đặt và kiểm thử

- Viết code để tạo một cây có gốc "Company".
- Thêm các phòng ban "HR", "IT", "Finance".
- Thêm nhân viên "Alice", "Bob" vào "HR".
- Thêm nhân viên "Charlie", "David" vào "IT".
- In toàn bộ cây.

### 4.2 Bài 2: Cài đặt duyệt BFS

Viết phương thức `traverse_bfs()` trong lớp Tree và kiểm tra kết quả.

### 4.3 Bài 3: Tìm kiếm bằng BFS

Viết phương thức `search_bfs(value)` để kiểm tra xem giá trị `value` có tồn tại trong cây hay không.

### 4.4 Bài 4: Tính chiều cao của cây

Viết phương thức `tree_height()` để tính chiều cao của cây.

# Tree data structure: Decision Tree

Hoàng-Nguyễn Vũ

## 1 Lý thuyết về cây quyết định

### 1.1 Cây quyết định là gì?

Cây quyết định (Decision Tree) là một mô hình học máy dựa trên cấu trúc cây để đưa ra quyết định hoặc dự đoán. Mỗi nút trong cây biểu diễn một điều kiện trên một thuộc tính, nhánh là kết quả của một điều kiện, và lá chứa giá trị dự đoán cuối cùng.

### 1.2 Ứng dụng của cây quyết định

- **Phân loại (Classification):** Xác định nhóm của một đối tượng dựa trên dữ liệu đầu vào.
- **Hồi quy (Regression):** Dự đoán giá trị số lượng dựa trên dữ liệu đầu vào.
- **Chẩn đoán y tế:** Xác định bệnh dựa trên triệu chứng.
- **Tài chính:** Dự đoán khả năng vỡ nợ của khách hàng.

## 2 Thước đo Gini trong cây quyết định

### 2.1 Thước đo Gini là gì?

Gini Index là một chỉ số đo độ tinh khiết của một tập dữ liệu. Nó giúp xác định mức độ hỗn loạn của tập dữ liệu và được sử dụng để chọn thuộc tính tốt nhất để chia nhánh trong cây quyết định.

### 2.2 Công thức tính Gini

Gini Index được tính bằng công thức:

$$Gini = 1 - \sum_{i=1}^n p_i^2$$

Trong đó:

- $p_i$  là tỷ lệ mẫu thuộc lớp  $i$  trong tập dữ liệu.
- $n$  là số lượng lớp.

Nếu một tập dữ liệu chỉ chứa một lớp duy nhất, Gini Index bằng 0 (hoàn toàn tinh khiết). Nếu tỷ lệ của các lớp là đều nhau, Gini Index đạt giá trị cao nhất.

### 2.3 Ví dụ tính Gini Index

Giả sử có tập dữ liệu phân loại như sau:

Tuổi	Mua sản phẩm?
22	Yes
35	No
26	Yes
45	No
30	Yes
40	No

Tỷ lệ của hai lớp:

- $p_{Yes} = \frac{3}{6} = 0.5$
- $p_{No} = \frac{3}{6} = 0.5$

$$Gini = 1 - (0.5^2 + 0.5^2) = 1 - (0.25 + 0.25) = 0.5$$

## 3 Cách sử dụng Gini trong cây quyết định

### 3.1 Các bước xây dựng cây quyết định dựa trên Gini

1. Tính Gini của tập dữ liệu ban đầu.
2. Chia tập dữ liệu thành các nhóm nhỏ dựa trên một thuộc tính.
3. Tính Gini trung bình có trọng số của các nhóm con.
4. Chọn thuộc tính có **Gini thấp nhất** để làm nút phân nhánh.
5. Lặp lại cho đến khi đạt điều kiện dừng.

## 4 Cài đặt cây quyết định sử dụng Gini

Dưới đây là một triển khai đơn giản của cây quyết định sử dụng Gini Index trong Python:

```

1 import numpy as np
2
3 def gini_index(groups, classes):
4     """Tính toán chỉ số Gini của một tập hợp"""
5     total_samples = float(sum([len(group) for group in groups]))
6     gini = 0.0
7     for group in groups:
8         size = float(len(group))
9         if size == 0:
10             continue
11         proportions = [sum(class_ == 1 for sample in group) / size
12                       for class_ in classes]
13         gini += sum(p * (1 - p) for p in proportions)
14
15     return gini / total_samples

```

```

11     score = sum([(row[-1] == c) for row in group for c in classes]) / size
12     gini += (1.0 - sum([score ** 2 for c in classes])) * (size / total_samples)
13     return gini
14
15 # Ví dụ dữ liệu
16 dataset = [
17     [2.8, 'Yes'],
18     [1.2, 'No'],
19     [3.6, 'Yes'],
20     [4.5, 'No'],
21     [5.1, 'Yes']
22 ]
23
24 # Chia tập dữ liệu theo một giá trị ngưỡng
25 def split_data(dataset, feature_index, threshold):
26     left = [row for row in dataset if row[feature_index] < threshold]
27     right = [row for row in dataset if row[feature_index] >= threshold]
28     return left, right
29
30 # Ví dụ tính Gini cho một cách chia dữ liệu
31 groups = split_data(dataset, 0, 3.0)
32 classes = ['Yes', 'No']
33 gini = gini_index(groups, classes)
34 print(f'Gini Index: {gini:.4f}')

```

## 5 Ví dụ: Xây dựng cây quyết định sử dụng OOP

### 5.1 Lớp TreeNode - Biểu diễn một nút trong cây

Chúng ta định nghĩa một lớp `TreeNode` để lưu trữ thông tin về các nút trong cây.

```

1 class TreeNode:
2     def __init__(self, feature_index=None, threshold=None, left=None,
3                  right=None, label=None):
4         """
5             Khởi tạo một nút trong cây quyết định.
6             - feature_index: Chỉ số thuộc tính được chọn để chia.
7             - threshold: Ngưỡng giá trị để phân chia dữ liệu.
8             - left: Nhánh trái của cây.
9             - right: Nhánh phải của cây.
10            - label: Nhận dự đoán nếu là nút lá.
11
12     self.feature_index = feature_index
13     self.threshold = threshold
14     self.left = left
15     self.right = right
16     self.label = label

```

## 5.2 Lớp DecisionTree - Xây dựng cây quyết định

Chúng ta xây dựng lớp `DecisionTree` với các phương thức chính để chia tập dữ liệu và xây dựng cây.

```

1 import numpy as np
2
3 class DecisionTree:
4     def __init__(self, max_depth=3):
5         """
6             Khởi tạo cây quyết định với độ sâu tối đa.
7         """
8         self.max_depth = max_depth
9         self.root = None
10
11    def gini_index(self, groups, classes):
12        """
13            Tính chỉ số Gini cho một tập hợp.
14        """
15        total_samples = sum([len(group) for group in groups])
16        gini = 0.0
17        for group in groups:
18            size = len(group)
19            if size == 0:
20                continue
21            score = 0.0
22            for class_val in classes:
23                proportion = [row[-1] for row in group].count(class_val) /
24                    size
25                score += proportion ** 2
26            gini += (1.0 - score) * (size / total_samples)
27        return gini
28
29    def split_data(self, dataset, feature_index, threshold):
30        """
31            Chia tập dữ liệu dựa trên một thuộc tính và ngưỡng giá trị.
32        """
33        left = [row for row in dataset if row[feature_index] < threshold]
34        right = [row for row in dataset if row[feature_index] >= threshold]
35        return left, right
36
37    def best_split(self, dataset):
38        """
39            Tìm thuộc tính tốt nhất để chia tập dữ liệu.
40        """
41        class_values = list(set(row[-1] for row in dataset))
42        best_index, best_threshold, best_score, best_groups = None, None,
43        float('inf'), None
44
45        for index in range(len(dataset[0]) - 1):
46            for row in dataset:
47                groups = self.split_data(dataset, index, row[index])
48                gini = self.gini_index(groups, class_values)

```

```
47             if gini < best_score:
48                 best_index, best_threshold, best_score, best_groups =
49                 index, row[index], gini, groups
50             return best_index, best_threshold, best_groups
51
52     def build_tree(self, dataset, depth=0):
53         """
54         Xây dựng cây quyết định đệ quy.
55         """
56
57         class_values = [row[-1] for row in dataset]
58
59         # Điều kiện dừng: Nếu chỉ có một lớp hoặc đạt đến độ sâu tối đa
60         if len(set(class_values)) == 1 or depth >= self.max_depth:
61             return TreeNode(label=max(set(class_values)), key=class_values.
62                             count())
63
64         # Tìm thuộc tính và giá trị ngưỡng tốt nhất để chia dữ liệu
65         feature_index, threshold, (left, right) = self.best_split(dataset)
66
67         # Nếu không thể chia tiếp, tạo nút lá
68         if not left or not right:
69             return TreeNode(label=max(set(class_values)), key=class_values.
70                             count())
71
72         # Xây dựng nhánh trái và nhánh phải
73         left_node = self.build_tree(left, depth + 1)
74         right_node = self.build_tree(right, depth + 1)
75
76         return TreeNode(feature_index, threshold, left_node, right_node)
77
78     def fit(self, dataset):
79         """
80         Huấn luyện cây quyết định bằng cách xây dựng cây từ dữ liệu đầu và
81         o.
82         """
83         self.root = self.build_tree(dataset)
84
85     def print_tree(self, node=None, depth=0):
86         """
87         In ra cây quyết định theo dạng phân cấp.
88         """
89         if node is None:
90             node = self.root
91
92             if node.label is not None:
93                 print(f'{    } * depth} [Leaf] Label: {node.label}')
94             else:
95                 print(f'{    } * depth} [Node] Feature {node.feature_index} <=
96 {node.threshold}')
97                 self.print_tree(node.left, depth + 1)
98                 self.print_tree(node.right, depth + 1)
```

### 5.3 Ví dụ sử dụng - Xây dựng cây từ tập dữ liệu

Chúng ta thử nghiệm xây dựng một cây quyết định với một tập dữ liệu đơn giản.

```

1 # Tập dữ liệu đơn giản: [Thuộc tính, Nhãn]
2 dataset = [
3     [2.8, 'Yes'],
4     [1.2, 'No'],
5     [3.6, 'Yes'],
6     [4.5, 'No'],
7     [5.1, 'Yes']
8 ]
9
10 # Khởi tạo và huấn luyện cây quyết định
11 tree = DecisionTree(max_depth=3)
12 tree.fit(dataset)
13
14 # In ra cây quyết định
15 print("Cây quyết định được xây dựng:")
16 tree.print_tree()
```

### 5.4 Kết quả mong đợi

Sau khi chạy đoạn mã trên, cây quyết định sẽ được hiển thị theo dạng phân cấp:

Cây quyết định được xây dựng:

```

[Node] Feature 0 <= 2.8
[Leaf] Label: No
[Node] Feature 0 <= 4.5
[Leaf] Label: Yes
[Node] Feature 0 <= 5.1
[Leaf] Label: No
[Leaf] Label: Yes
```

## 6 Bài tập thực hành

### Bài 1: Tính chỉ số Gini cho một tập dữ liệu

Tính toán chỉ số Gini cho tập dữ liệu sau:

Lương	Đủ điều kiện vay vốn?
50	Yes
20	No
30	No
70	Yes
40	No
60	Yes

## Bài 2: Mở rộng cây quyết định

Tập dữ liệu sau mô phỏng thông tin của một học viên có đam mê với toán, hội họa, tiếng anh để khảo sát họ có thích AI hay không

	love_math	love_art	love_english	love_ai
0	yes	yes	no	no
1	yes	no	no	no
2	no	yes	yes	yes
3	no	yes	yes	yes
4	yes	yes	yes	yes
5	yes	no	yes	no
6	no	no	yes	no

Bảng 1: Bảng dữ liệu về sở thích của học sinh

Hãy sử dụng thước đo Gini, và áp dụng kỹ thuật xây dựng cây quyết định với bộ data trên với thước đo Gini

Cây quyết định được xây dựng:

```
[Node] Feature love_art <= yes
Left branch (<=):
[Leaf] Label love_ai: no
Right branch (>):
[Node] Feature love_english <= yes
Left branch (<=):
[Leaf] Label love_ai: no
Right branch (>):
[Leaf] Label love_ai: yes
```

# Phân loại bệnh cúm bằng Naive Bayes

*Hoàng-Nguyễn Vũ*

## 1 Lý thuyết

Naive Bayes là một thuật toán phân loại dựa trên định lý Bayes với giả định độc lập có điều kiện giữa các đặc trưng. Công thức chính của định lý Bayes được biểu diễn như sau:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \quad (1)$$

Trong đó:

- $P(Y|X)$ : Xác suất xảy ra lớp  $Y$  (bị cúm hay không) khi biết đặc trưng  $X$  (triệu chứng bệnh nhân).
- $P(X|Y)$ : Xác suất có triệu chứng  $X$  khi đã biết bệnh nhân thuộc lớp  $Y$ .
- $P(Y)$ : Xác suất tiên nghiệm của lớp  $Y$ .
- $P(X)$ : Xác suất xảy ra triệu chứng  $X$ .

Trong mô hình Naive Bayes, ta giả định các đặc trưng là độc lập có điều kiện, do đó xác suất có điều kiện có thể được viết thành tích của các xác suất riêng lẻ:

$$P(X|Y) = P(X_1|Y) \cdot P(X_2|Y) \cdot \dots \cdot P(X_n|Y) \quad (2)$$

Tuy nhiên, một vấn đề xảy ra khi có một đặc trưng nào đó chưa từng xuất hiện trong dữ liệu huấn luyện, dẫn đến xác suất có điều kiện bằng 0. Điều này khiến kết quả cuối cùng của  $P(Y|X)$  trở thành 0. Để khắc phục vấn đề này, ta sử dụng kỹ thuật Laplace Smoothing, được tính theo công thức:

$$P(X|Y) = \frac{\text{count}(X, Y) + k}{\text{count}(Y) + m \cdot k} \quad (3)$$

Trong đó:

- $\text{count}(X, Y)$ : Số lần xuất hiện của đặc trưng  $X$  trong tập hợp mẫu thuộc lớp  $Y$ .
- $\text{count}(Y)$ : Số lần xuất hiện của lớp  $Y$  trong dữ liệu huấn luyện.
- $m$ : Số lượng giá trị khả dĩ của biến  $X$ .
- $k$ : Hằng số điều chỉnh (thường lấy  $k = 1$ ) để tránh xác suất bằng 0.

Laplace Smoothing giúp đảm bảo rằng mọi xác suất có điều kiện đều có giá trị hợp lệ và mô hình hoạt động ổn định hơn khi gặp dữ liệu mới.

## 2 Bài tập

Dữ liệu từ 10 bệnh nhân được ghi lại như sau:

Sốt	Ho	Đau họng	Mệt mỏi	Flu
Cao	Có	Có	Có	Có
Cao	Không	Có	Không	Có
Cao	Có	Không	Có	Có
Thấp	Có	Có	Có	Không
Cao	Có	Có	Không	Có
Thấp	Không	Có	Không	Không
Thấp	Có	Không	Có	Không
Cao	Không	Có	Có	Có
Thấp	Không	Không	Không	Không
Cao	Có	Có	Có	Có

Bệnh nhân mới có các triệu chứng: Sốt = Cao, Ho = Có, Đau họng = Có, Mệt mỏi = Không. Hãy tính xác suất để bệnh nhân này bị cúm bằng công thức Naive Bayes và kết luận bệnh nhân có khả năng bị cúm hay không.

```

1 import numpy as np
2 import pandas as pd
3
4 # Dữ liệu huấn luyện
5 # Tạo DataFrame chứa dữ liệu bệnh nhân
6 # Your code here
7
8 data = pd.DataFrame({
9     "Sốt": ["Cao", "Cao", "Cao", "Thấp", "Cao", "Thấp", "Thấp", "Cao", "Thấp", "Cao"],
10    "Ho": ["Có", "Không", "Có", "Có", "Có", "Không", "Có", "Không", "Có", "Có"],
11    "Đau họng": ["Có", "Có", "Không", "Có", "Có", "Có", "Không", "Có", "Không", "Có"],
12    "Mệt mỏi": ["Có", "Không", "Có", "Có", "Không", "Không", "Có", "Có", "Không", "Có"],
13    "Flu": ["Có", "Có", "Có", "Không", "Có", "Không", "Không", "Có", "Không", "Có"]
14 })
15
16 # Dữ liệu bệnh nhân cần dự đoán
17 new_patient = {"Sốt": "Cao", "Ho": "Có", "Đau họng": "Có", "Mệt mỏi": "Không"}
18
19 # Tính xác suất tiên nghiệm P(C)
20 P_flu = data["Flu"].value_counts(normalize=True)[("Có")] # P(C=1)
21 P_not_flu = 1 - P_flu # P(C=0)
22
23 # Hàm tính xác suất có điều kiện P(x_i|C) với Laplace smoothing

```

```

24 def conditional_prob(feature, value, flu_status):
25     # Your code here
26
27 # Tính  $P(X|C) = P(x_1|C) * P(x_2|C) * \dots * P(x_n|C)$  theo Naive Bayes
28 P_X_given_flu = ??? #  $P(X|C=1)$ 
29 P_X_given_not_flu = ??? #  $P(X|C=0)$ 
30
31 for feature, value in new_patient.items():
32     # Your code here
33
34 # Tính  $P(C|X)$  trực tiếp từ tỷ lệ (không cần  $P(X)$ )
35 numerator_flu = P_X_given_flu * P_flu
36 numerator_not_flu = P_X_given_not_flu * P_not_flu
37
38 # Chuẩn hóa xác suất
39 total = numerator_flu + numerator_not_flu
40 P_flu_given_X = numerator_flu / total
41 P_not_flu_given_X = numerator_not_flu / total
42
43 print("\nKết quả Naive Bayes:")
44 print(f"P(Flu=Có|X) = {P_flu_given_X:.4f}")
45 print(f"P(Flu=Không|X) = {P_not_flu_given_X:.4f}")
46
47 # Kết luận
48 print("\nKết luận:", "Bệnh nhân có khả năng bị cúm." if P_flu_given_X >
    P_not_flu_given_X else "Bệnh nhân KHÔNG có khả năng bị cúm.")

```

$P(Sốt=Cao|Flu=Có) = 0.8750$   
 $P(Sốt=Cao|Flu=Không) = 0.1667$   
 $P(Ho=Có|Flu=Có) = 0.6250$   
 $P(Ho=Có|Flu=Không) = 0.5000$   
 $P(Đau họng=Có|Flu=Có) = 0.7500$   
 $P(Đau họng=Có|Flu=Không) = 0.5000$   
 $P(Mệt mỏi=Không|Flu=Có) = 0.3750$   
 $P(Mệt mỏi=Không|Flu=Không) = 0.5000$

Kết quả Naive Bayes:

$P(Flu=Có|X) = 0.9172$   
 $P(Flu=Không|X) = 0.0828$

Kết luận: Bệnh nhân có khả năng bị cúm.

# Calculus in Machine Learning - Basic Derivatives

*Hoàng-Nguyễn Vũ*

## 1 Lý thuyết: Đạo hàm và phương pháp sai phân hữu hạn

### 1.1 Định nghĩa đạo hàm

Đạo hàm của một hàm số  $f(x)$  tại một điểm  $x$  được định nghĩa là:

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}$$

Đạo hàm mô tả tốc độ thay đổi của hàm số tại một điểm và có nhiều ứng dụng quan trọng trong:

- Tìm cực trị của hàm số.
- Tính toán độ dốc của đồ thị hàm số.
- Giải quyết các bài toán tối ưu hóa trong khoa học dữ liệu và học máy.

### 1.2 Phương pháp sai phân hữu hạn

Trong lập trình, ta không thể tính giới hạn khi  $h \rightarrow 0$ , do đó ta xấp xỉ đạo hàm bằng phương pháp sai phân hữu hạn:

$$f'(x) \approx \frac{f(x + h) - f(x)}{h}$$

với  $h$  là một số rất nhỏ (ví dụ:  $h = 10^{-5}$ ).

### 1.3 Bảng công thức đạo hàm

#### 1.3.1 Các đạo hàm cơ bản

Hàm số $f(x)$	Đạo hàm $f'(x)$
$c$ (hằng số)	0
$x^n$	$nx^{n-1}$
$e^x$	$e^x$
$a^x$	$a^x \ln a$
$\ln x$	$\frac{1}{x}$
$\log_a x$	$\frac{1}{x \ln a}$
$\sin x$	$\cos x$
$\cos x$	$-\sin x$
$\tan x$	$\sec^2 x$
$\cot x$	$-\csc^2 x$
$\sec x$	$\sec x \tan x$
$\csc x$	$-\csc x \cot x$
$\sqrt{x}$	$\frac{1}{2\sqrt{x}}$

#### 1.3.2 Công thức đạo hàm với $u(x)$ là một hàm số

Công thức	Đạo hàm
$(u + v)'$	$u' + v'$
$(u - v)'$	$u' - v'$
$(uv)'$	$u'v + uv'$
$\left(\frac{u}{v}\right)'$	$\frac{u'v - uv'}{v^2}$
$(u^n)'$	$n u^{n-1} u'$
$e^u$	$e^u u'$
$a^u$	$a^u \ln a \cdot u'$
$\ln u$	$\frac{u'}{u}$
$\log_a u$	$\frac{u'}{u \ln a}$
$\sin u$	$\cos u \cdot u'$
$\cos u$	$-\sin u \cdot u'$
$\tan u$	$\sec^2 u \cdot u'$

## 2 Bài tập: Lập trình tính đạo hàm bằng Python

Dưới đây là 8 bài tập yêu cầu viết chương trình Python sử dụng `numpy` và `matplotlib` để tính đạo hàm số học bằng phương pháp sai phân hữu hạn và vẽ đồ thị của hàm số cùng với đạo hàm của nó.

### 2.1 Bài 1: Đạo hàm của hàm bậc hai

Tính đạo hàm số học và vẽ đồ thị của:

$$f(x) = x^2 - 3x + 2$$

## 2.2 Bài 2: Đạo hàm của hàm bậc ba

Tính đạo hàm số học và vẽ đồ thị của:

$$f(x) = x^3 - 3x^2 + 4x - 2$$

## 2.3 Bài 3: Đạo hàm của hàm số mũ

Tính đạo hàm số học và vẽ đồ thị của:

$$f(x) = e^x$$

## 2.4 Bài 4: Đạo hàm của hàm logarit

Tính đạo hàm số học và vẽ đồ thị của:

$$f(x) = \ln(x)$$

Lưu ý chọn khoảng giá trị  $x > 0$  để đảm bảo tính toán hợp lệ.

## 2.5 Bài 5: Đạo hàm của hàm $\sin(x)$

Tính đạo hàm số học và vẽ đồ thị của:

$$f(x) = \sin(x)$$

## 2.6 Bài 6: Đạo hàm của hàm $\cos(x)$

Tính đạo hàm số học và vẽ đồ thị của:

$$f(x) = \cos(x)$$

## 2.7 Bài 7: Đạo hàm của hàm $\tan(x)$

Tính đạo hàm số học và vẽ đồ thị của:

$$f(x) = \tan(x)$$

Lưu ý tránh các điểm gián đoạn tại  $x = \frac{\pi}{2} + k\pi$  với  $k$  là số nguyên.

## 2.8 Bài 8: Đạo hàm của hàm căn bậc hai

Tính đạo hàm số học và vẽ đồ thị của:

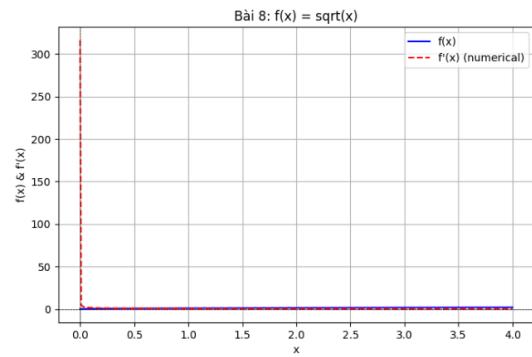
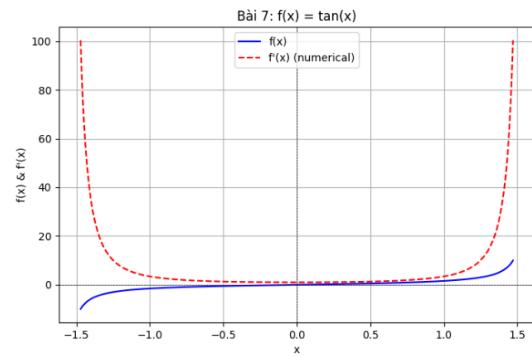
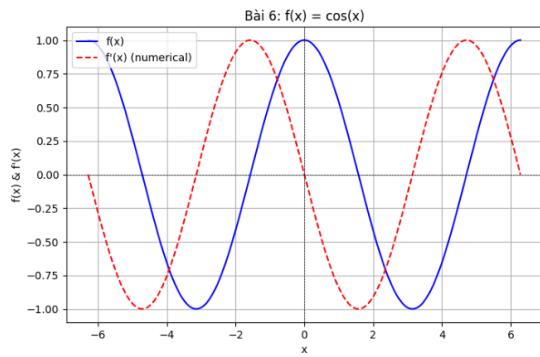
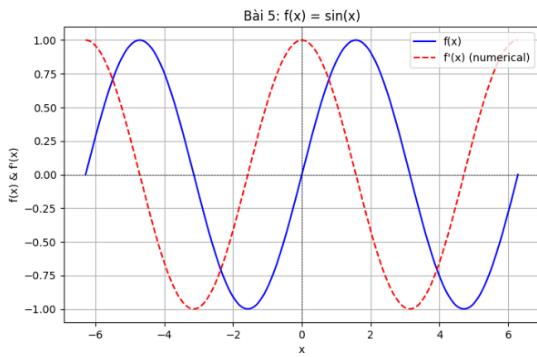
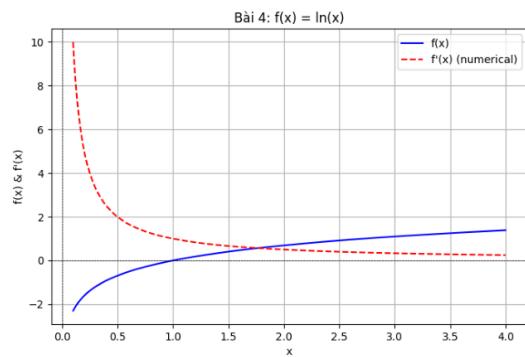
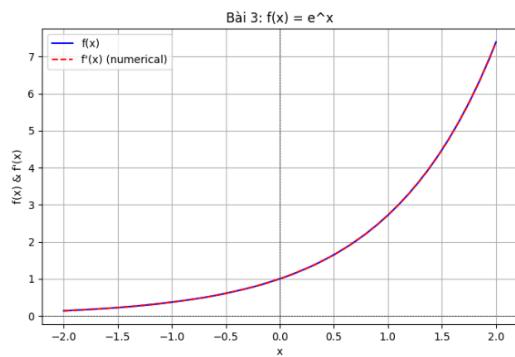
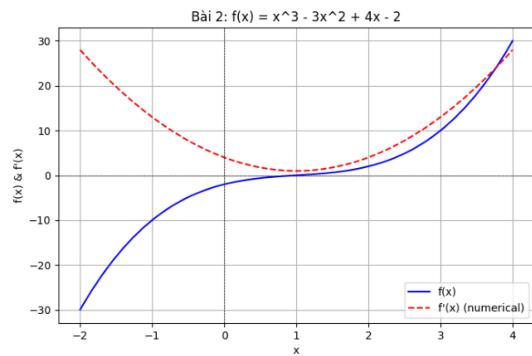
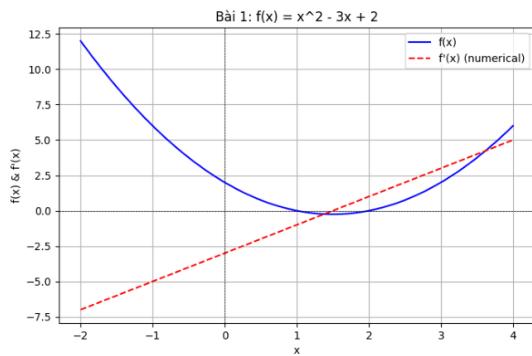
$$f(x) = \sqrt{x}$$

Lưu ý chỉ xét với  $x \geq 0$ .

### 3 Hướng dẫn chung

- Viết một hàm số  $f(x)$ .
- Sử dụng phương pháp sai phân hữu hạn để tính đạo hàm.
- Vẽ đồ thị của  $f(x)$  và  $f'(x)$  bằng matplotlib.
- Phân tích các điểm cực trị hoặc gián đoạn (nếu có).

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # Hàm tính đạo hàm bằng sai phân hữu hạn
5 def numerical_derivative(f, x, h=1e-5):
6     # Your code here #
7
8 # Hàm vẽ đồ thị
9 def plot_function_and_derivative(f, x_range=(-2, 2), title="Đồ thị f(x) và
10    f'(x)"):
11    x_vals = np.linspace(x_range[0], x_range[1], 400)
12    y_vals = f(x_vals)
13    y_derivative_vals = numerical_derivative(f, x_vals)
14
15    plt.figure(figsize=(8, 5))
16    plt.plot(x_vals, y_vals, label="f(x)", color='blue')
17    plt.plot(x_vals, y_derivative_vals, label="f'(x) (numerical)", color='red',
18             linestyle="--")
19    plt.axhline(0, color='black', linewidth=0.5, linestyle='--')
20    plt.axvline(0, color='black', linewidth=0.5, linestyle='--')
21    plt.xlabel("x")
22    plt.ylabel("f(x) & f'(x)")
23    plt.title(title)
24    plt.legend()
25    plt.grid()
26    plt.show()
```



# Calculus in Machine Learning - Basic Chain rule

*Hoàng-Nguyễn Vũ*

## 1 Lý thuyết: Đạo hàm của hàm hợp

### 1.1 Định nghĩa đạo hàm của hàm hợp

Cho hai hàm số  $u = g(x)$  và  $y = f(u)$ , khi đó  $y$  là một hàm hợp của  $x$ :

$$y = f(g(x))$$

Đạo hàm của  $y$  theo  $x$  được tính bằng công thức:

$$\frac{d}{dx}f(g(x)) = f'(g(x)) \cdot g'(x)$$

Công thức này còn được gọi là **quy tắc đạo hàm của hàm hợp**.

### 1.2 Ý nghĩa của đạo hàm của hàm hợp

- Khi một hàm số được biểu diễn dưới dạng  $f(g(x))$ , đạo hàm của nó phản ánh tốc độ thay đổi của  $f$  theo  $x$ , thông qua tác động của  $g(x)$ . - Công thức này đặc biệt hữu ích khi làm việc với các hàm số phức hợp như  $e^{x^2}$ ,  $\ln(x^3 + 1)$ , hoặc  $\sin(x^2 - 2x)$ .

### 1.3 Ví dụ minh họa

**Ví dụ 1:** Tính đạo hàm của hàm số:

$$f(x) = (3x^2 + 5)^4$$

**Lời giải:** Gọi  $g(x) = 3x^2 + 5$ , khi đó  $f(x) = g(x)^4$ . Theo quy tắc đạo hàm của hàm hợp:

$$f'(x) = 4g(x)^3 \cdot g'(x)$$

Với  $g'(x) = 6x$ , ta có:

$$f'(x) = 4(3x^2 + 5)^3 \cdot 6x = 24x(3x^2 + 5)^3$$

**Ví dụ 2:** Tính đạo hàm của hàm số:

$$f(x) = e^{\sin x}$$

**Lời giải:** Gọi  $g(x) = \sin x$ , khi đó  $f(x) = e^{g(x)}$ . Theo quy tắc đạo hàm của hàm hợp:

$$f'(x) = e^{\sin x} \cdot \cos x$$

## 1.4 Công thức đạo hàm của một số hàm hợp thường gặp

Hàm số $f(g(x))$	Đạo hàm $f'(g(x)) \cdot g'(x)$
$\sin(g(x))$	$\cos(g(x)) \cdot g'(x)$
$\cos(g(x))$	$-\sin(g(x)) \cdot g'(x)$
$\tan(g(x))$	$\sec^2(g(x)) \cdot g'(x)$
$\ln(g(x))$	$\frac{g'(x)}{g(x)}$
$e^{g(x)}$	$e^{g(x)} \cdot g'(x)$
$\sqrt{g(x)}$	$\frac{g'(x)}{2\sqrt{g(x)}}$

## 2 Bài tập: Đạo hàm của hàm hợp

Dưới đây là các bài tập yêu cầu tính đạo hàm của các hàm hợp bằng phương pháp giải tích. Sau khi tính toán, hãy kiểm tra lại bằng cách lập trình Python sử dụng phương pháp sai phân hữu hạn.

### Bài 1: Đạo hàm của hàm mũ hợp

Tính đạo hàm của hàm số sau:

$$f(x) = e^{x^3 - 2x + 1}$$

Hãy tìm các điểm mà đạo hàm bằng 0.

### Bài 2: Đạo hàm của hàm logarit hợp

Tính đạo hàm của:

$$f(x) = \ln(5x^2 + 3x + 1)$$

Xác định khoảng mà đạo hàm dương hoặc âm.

### Bài 3: Đạo hàm của hàm căn bậc hai hợp

Tính đạo hàm của:

$$f(x) = \sqrt{x^4 + 2x^2 + 1}$$

Hãy phân tích sự thay đổi của đạo hàm trên tập xác định.

### Bài 4: Đạo hàm của hàm lượng giác hợp

Tính đạo hàm của:

$$f(x) = \sin(x^2 - 3x + 2)$$

Xác định điểm mà đạo hàm bằng 0 và tìm các điểm cực trị.

## Bài 5: Đạo hàm của hàm hợp có nhiều lớp

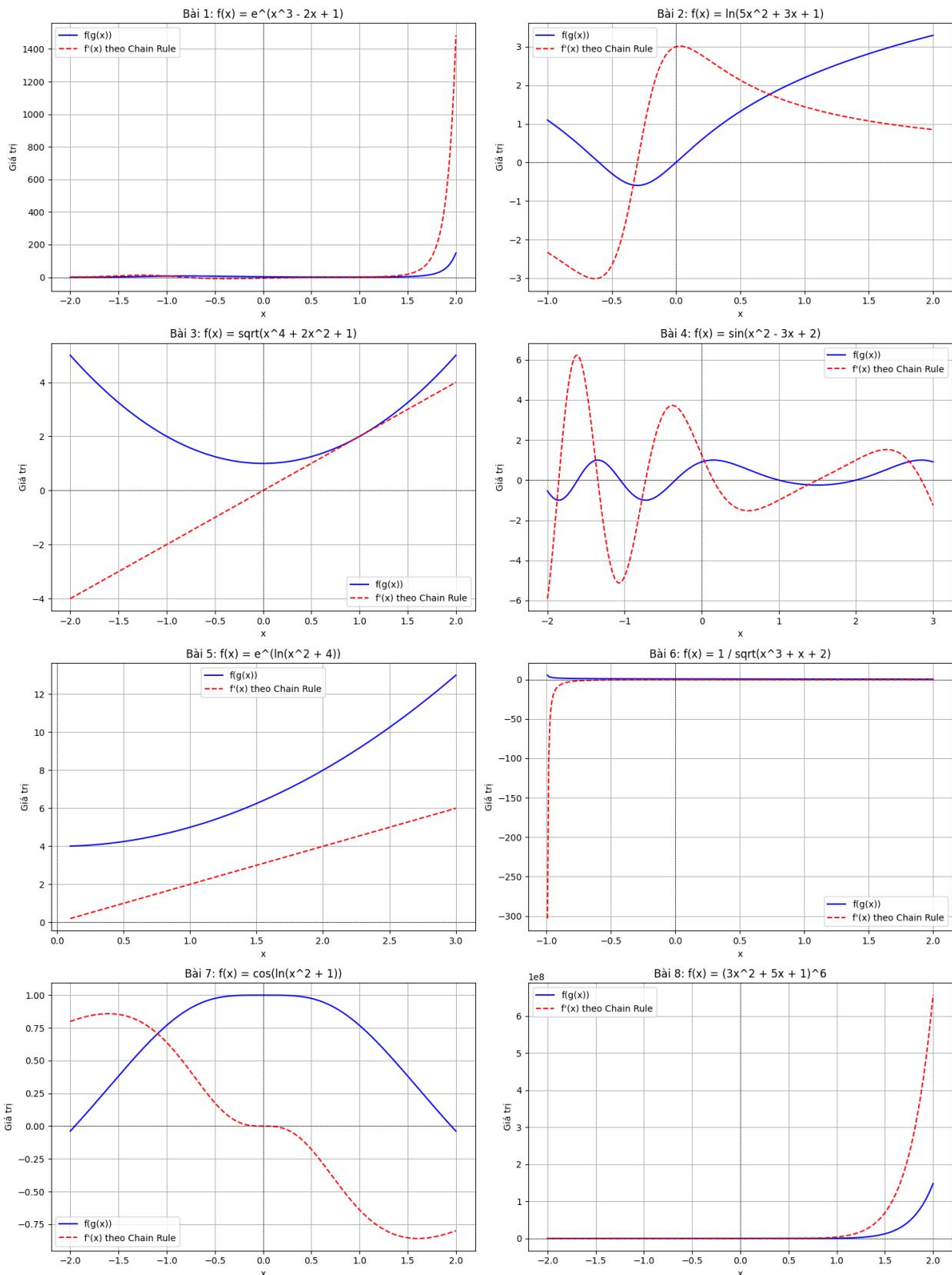
Tính đạo hàm của:

$$f(x) = e^{\ln(x^2+4)}$$

So sánh kết quả với đạo hàm của  $g(x) = x^2 + 4$  để thấy mối quan hệ giữa chúng.

```

1 # Hàm tính đạo hàm bằng sai phân hữu hạn
2 def numerical_derivative(f, x, h=1e-5):
3     return (f(x + h) - f(x)) / h
4
5 # Hàm vẽ đồ thị của hàm số và đạo hàm
6 def plot_chain_rule(f, g, x_range=(-2, 2), title="Đạo hàm Chain Rule"):
7     x_vals = np.linspace(x_range[0], x_range[1], 400)
8
9     # Tính giá trị của g(x) và f(g(x))
10    g_vals = g(x_vals)
11    f_vals = f(g_vals)
12
13    # Tính đạo hàm theo Chain Rule
14    # Your Code here #
15
16    # Vẽ đồ thị của f(g(x)) và đạo hàm
17    plt.figure(figsize=(8, 5))
18    plt.plot(x_vals, f_vals, label="f(g(x))", color='blue')
19    plt.plot(x_vals, chain_rule_derivative_vals, label="f'(x) theo Chain
20    Rule", color='red', linestyle="--")
21
22    plt.axhline(0, color='black', linewidth=0.5, linestyle='--')
23    plt.axvline(0, color='black', linewidth=0.5, linestyle='--')
24
25    plt.xlabel("x")
26    plt.ylabel("Giá trị")
27    plt.title(title)
28    plt.legend()
29    plt.grid()
```



# Calculus in Machine Learning - Derivatives in Artificial Intelligence

*Hoàng-Nguyễn Vũ*

## 1 Giới thiệu về Giải tích trong AI

Giải tích là một công cụ quan trọng trong trí tuệ nhân tạo (AI), đặc biệt trong lĩnh vực học máy (Machine Learning) và học sâu (Deep Learning). Một số ứng dụng quan trọng của giải tích trong AI:

- Tối ưu hóa mô hình bằng Gradient Descent.
- Tính đạo hàm để cập nhật tham số trong học sâu.
- Tính đạo hàm riêng để huấn luyện mạng nơ-ron bằng thuật toán Backpropagation.
- Tích phân trong xác suất và thống kê (phân bố xác suất).

## 2 Đạo hàm trong học máy

### 2.1 Định nghĩa đạo hàm và ý nghĩa trong AI

Đạo hàm đo lường tốc độ thay đổi của một hàm số. Trong AI, đạo hàm được sử dụng để điều chỉnh tham số mô hình để giảm lỗi dự đoán.

Cho một hàm  $f(x)$ , đạo hàm của nó được định nghĩa là:

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}$$

Ví dụ trong AI, nếu ta có hàm mất mát (Loss Function)  $J(\theta)$ , đạo hàm của nó giúp ta cập nhật tham số  $\theta$  để làm giảm giá trị mất mát:

$$\frac{dJ}{d\theta}$$

### 2.2 Gradient Descent - Tối ưu hóa mô hình

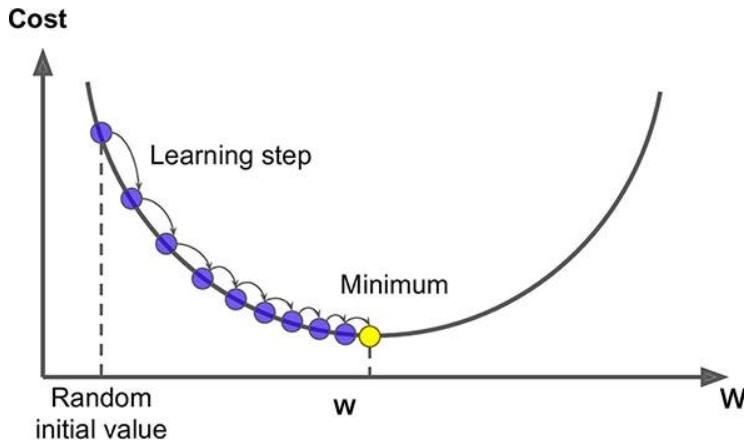
Gradient Descent là thuật toán tối ưu hóa quan trọng giúp cập nhật tham số của mô hình dựa trên gradient của hàm mất mát. Công thức cập nhật tham số:

$$\theta_{t+1} = \theta_t - \alpha \nabla J(\theta)$$

trong đó:

+  $\alpha$  là tốc độ học (learning rate).

+  $\nabla J(\theta)$  là gradient của hàm mất mát.



Hình 1: Minh họa Gradient Descent

### 3 Đạo hàm của hàm hợp và Backpropagation

#### 3.1 Đạo hàm của hàm hợp trong học sâu

Trong mạng nơ-ron nhân tạo (Neural Network), đầu ra của một lớp là đầu vào của lớp tiếp theo. Khi huấn luyện mô hình, ta cần tính toán đạo hàm của hàm hợp nhiều lớp.

Giả sử mạng có các lớp tuần tự:

$$a^{(l+1)} = f(W^{(l)}a^{(l)} + b^{(l)})$$

Đạo hàm của hàm mất mát  $L$  theo trọng số  $W$  được tính bằng quy tắc Chain Rule:

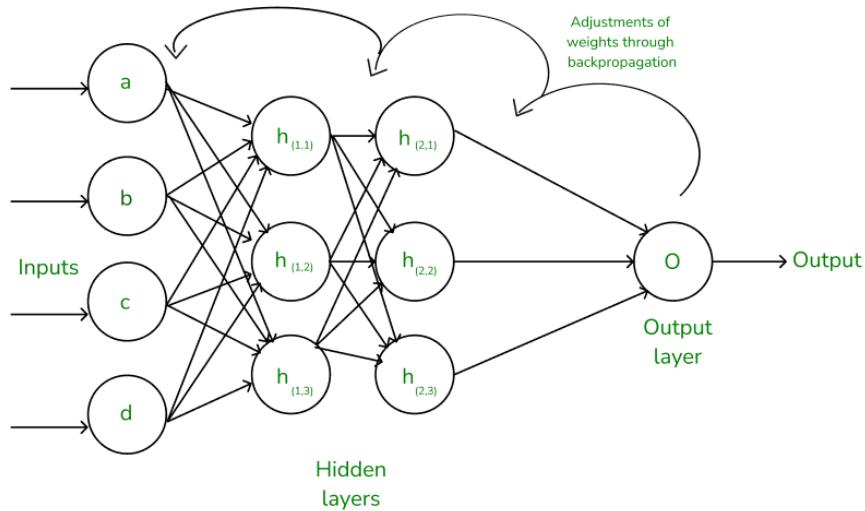
$$\frac{dL}{dW} = \frac{dL}{da^{(l+1)}} \cdot \frac{da^{(l+1)}}{d(W^{(l)}a^{(l)} + b^{(l)})} \cdot \frac{d(W^{(l)}a^{(l)} + b^{(l)})}{dW}$$

#### 3.2 Thuật toán Backpropagation - Lan truyền ngược

Backpropagation sử dụng Chain Rule để tính gradient từ đầu ra về lại các lớp trước để cập nhật trọng số. Công thức tổng quát:

$$\delta^{(l)} = (W^{(l)})^T \delta^{(l+1)} \circ f'(z^{(l)})$$

trong đó: -  $\delta^{(l)}$  là gradient của lớp  $l$ . -  $f'(z)$  là đạo hàm của hàm kích hoạt.



Hình 2: Minh họa thuật toán Backpropagation

## 4 Bài tập thực hành

### Bài 1: Đạo Hàm của Hàm Kích Hoạt Sigmoid

Hàm Sigmoid thường được sử dụng trong các mạng nơ-ron nhân tạo để chuẩn hóa giá trị đầu vào thành một giá trị giữa 0 và 1:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Đạo hàm của Sigmoid được tính như sau:

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

**Dữ liệu mẫu:** Cho  $x = \{-2, -1, 0, 1, 2\}$ , hãy tính  $\sigma(x)$  và  $\sigma'(x)$ .

```

1 import numpy as np
2
3 def sigmoid(x):
4     # Your code here #
5
6 def sigmoid_derivative(x):
7     # Your code here #
8
9 x = np.array([-2, -1, 0, 1, 2])
10 print("Sigmoid:", sigmoid(x))
11 print("Sigmoid derivative:", sigmoid_derivative(x))

```

### Bài 2: Đạo Hàm của Hàm Tổn Thất MSE

Hàm tổn thất trung bình bình phương (MSE) được định nghĩa như sau:

$$L(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Đạo hàm của MSE theo  $\hat{y}$ :

$$\frac{\partial L}{\partial \hat{y}_i} = -\frac{2}{n}(y_i - \hat{y}_i)$$

Dữ liệu mẫu:

$$y = \{3, 5, 2, 8\}, \quad \hat{y} = \{2.5, 4.8, 2.1, 7.5\}$$

```

1 def mse_derivative(y_true, y_pred):
2     # Your code here #
3
4 y_true = np.array([3, 5, 2, 8])
5 y_pred = np.array([2.5, 4.8, 2.1, 7.5])
6 print("Gradient của MSE:", mse_derivative(y_true, y_pred))

```

### Bài 3: Đạo Hàm của Hàm Kích Hoạt ReLU

Hàm ReLU được định nghĩa như sau:

$$ReLU(x) = \max(0, x)$$

Đạo hàm của ReLU:

$$ReLU'(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

Dữ liệu mẫu:  $x = \{-2, -1, 0, 1, 2\}$ .

```

1 def relu(x):
2     # Your code here #
3
4 def relu_derivative(x):
5     # Your code here #
6
7 x = np.array([-2, -1, 0, 1, 2])
8 print("ReLU:", relu(x))
9 print("ReLU derivative:", relu_derivative(x))

```

### Bài 4: Đạo Hàm của Hàm Softmax và Cross-Entropy

Hàm Softmax chuyển đổi một vector thành một phân phối xác suất:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Đạo hàm của Softmax:

$$\frac{\partial \sigma_i}{\partial z_j} = \sigma_i(1 - \sigma_i) \quad \text{nếu } i = j$$

Dữ liệu mẫu:

$$z = \{2.0, 1.0, 0.1\}$$

```

1 def softmax(z):
2     # Your code here #
3
4 def softmax_derivative(z):
5     # Your code here #
6
7 z = np.array([2.0, 1.0, 0.1])
8 print("Softmax:", softmax(z))
9 print("Gradient của Softmax:", softmax_derivative(z))

```

## Bài 5: Đạo Hàm của Hàm Kích Hoạt Tanh

Hàm tanh được định nghĩa như sau:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Đạo hàm của nó:

$$\tanh'(x) = 1 - \tanh^2(x)$$

Dữ liệu mẫu:  $x = \{-2, -1, 0, 1, 2\}$ .

```

1 def tanh(x):
2     # Your code here #
3
4 def tanh_derivative(x):
5     # Your code here #
6
7 x = np.array([-2, -1, 0, 1, 2])
8 print("Tanh:", tanh(x))
9 print("Tanh derivative:", tanh_derivative(x))

```

# Calculus in Machine Learning - Optimization

Hoàng-Nguyễn Vũ

## 1 Lý Thuyết về Optimization

### 1.1 Khái niệm tối ưu hóa

Tối ưu hóa (Optimization) là quá trình tìm giá trị tối ưu của một hàm mục tiêu, thường là cực tiểu hoặc cực đại của hàm số. Bài toán tối ưu hóa tổng quát có dạng:

$$\min_x f(x) \quad \text{subject to constraints}$$

trong đó:

- $f(x)$  là hàm mục tiêu cần tối ưu.
- $x$  là biến quyết định.
- Constraints (ràng buộc) giới hạn miền giá trị hợp lệ của  $x$ .

### 1.2 Các phương pháp tối ưu hóa phổ biến

- **Gradient Descent:** Sử dụng đạo hàm để cập nhật giá trị  $x$  theo công thức:

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

với  $\alpha$  là tốc độ học (learning rate).

- **Newton's Method:** Tăng tốc hội tụ bằng cách sử dụng đạo hàm bậc 2:

$$x_{t+1} = x_t - \frac{f'(x_t)}{f''(x_t)}$$

- **Linear Programming (Quy hoạch tuyến tính - Đọc thêm):** Dùng để tối ưu bài toán với ràng buộc tuyến tính, thường sử dụng thư viện như `scipy.optimize.linprog`.

## 2 Bài Tập

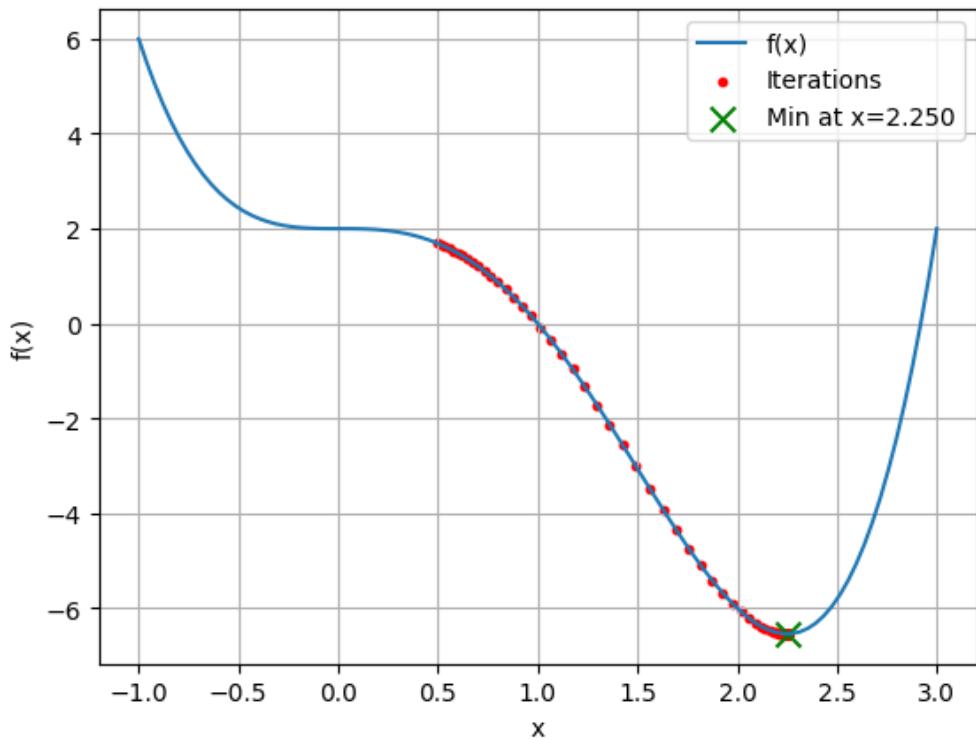
### Bài 1: Tối ưu hàm một biến bằng Gradient Descent

Tìm cực tiểu của hàm số:

$$f(x) = x^4 - 3x^3 + 2$$

**Giải pháp:** Dùng Gradient Descent với đạo hàm:

$$f'(x) = 4x^3 - 9x^2$$



```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def f(x):
5     # Your code here #
6
7 def f_prime(x):
8     # Your code here #
9
10 # Gradient Descent
11 def gradient_descent(f_prime, x_init, learning_rate=0.01, epochs=100):
12     x = x_init
13     history = [x]
14
15     for _ in range(epochs):
16         # Your code here #
17
18     return x, history
19
20 # Chạy Gradient Descent
21 x_init = 0.5
22 learning_rate = 0.01
23 epochs = 100
24 x_optimal, history = gradient_descent(f_prime, x_init, learning_rate,
25                                         epochs)
26
27 # Vẽ đồ thị
28 x_vals = np.linspace(-1, 3, 100)
```

```

28 y_vals = f(x_vals)
29
30 plt.plot(x_vals, y_vals, label="f(x)")
31 plt.scatter(history, [f(x) for x in history], color="red", s=10, label="Iterations")
32 plt.scatter(x_optimal, f(x_optimal), color="green", marker="x", s=100,
33             label=f"Min at x={x_optimal:.3f}")
34 plt.xlabel("x")
35 plt.ylabel("f(x)")
36 plt.legend()
37 plt.grid()
38 plt.show()

```

## Bài 2: Hồi quy tuyến tính với Gradient Descent

Tìm hệ số  $w, b$  tối ưu cho mô hình hồi quy:

$$y = wx + b$$

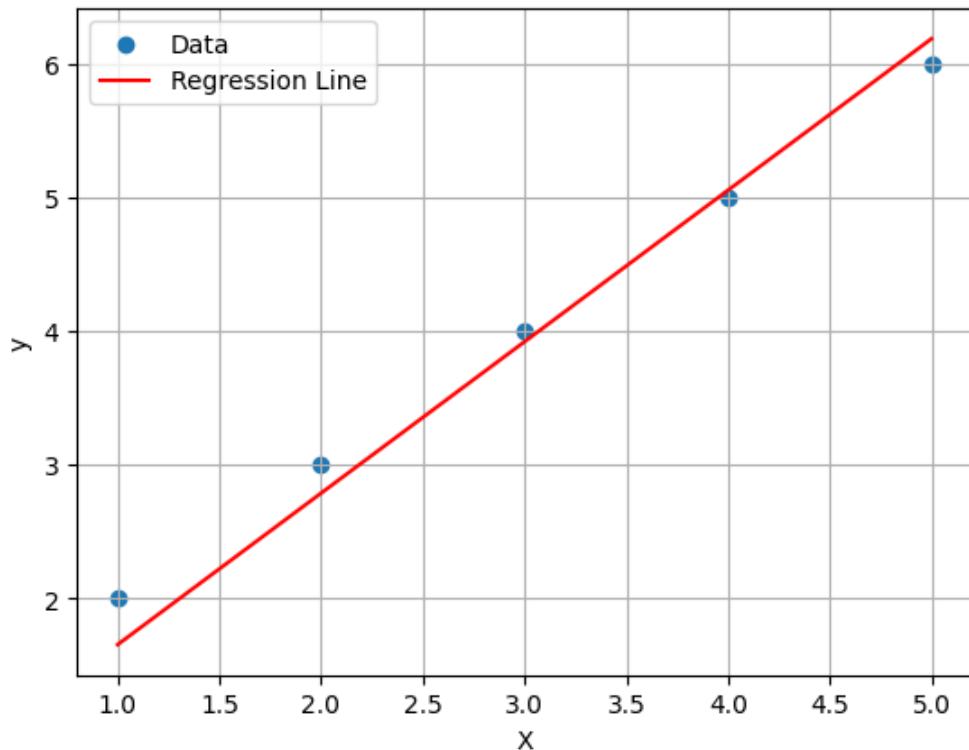
Dữ liệu mẫu:

$X$	$y$
1	2
2	3
3	4
4	5
5	6

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 X = np.array([1, 2, 3, 4, 5])
5 y = np.array([2, 3, 4, 5, 6])
6
7 w, b = 0.0, 0.0
8 learning_rate = 0.01
9 epochs = 1000
10
11 for _ in range(epochs):
12     # Your code here #
13
14 # Vẽ đồ thị
15 plt.scatter(X, y, label="Data")
16 plt.plot(X, y_pred, color='red', label="Regression Line")
17 plt.xlabel("X")
18 plt.ylabel("y")
19 plt.legend()
20 plt.grid()
21 plt.show()
22
23 print(f"Hệ số tối ưu: w = {w:.3f}, b = {b:.3f}")

```



### Bài 3: Newton's Method

Tìm cực tiểu của hàm số:

$$f(x) = x^3 - 6x^2 + 4x + 12$$

Dùng Newton's Method với đạo hàm:

$$f'(x) = 3x^2 - 12x + 4, \quad f''(x) = 6x - 12$$

```

1 def f_prime(x):
2     # Your code here #
3
4 def f_double_prime(x):
5     # Your code here #
6
7 x = 5
8 for _ in range(10):
9     # Your code here #
10
11 print(f"Cực tiểu tại x = {x:.3f}")
12 ## Đáp số: Cực tiểu tại x = 3.633

```

# Dự đoán giá vàng bằng Hồi quy tuyến tính

*Hoàng-Nguyễn Vũ*

## 1 Mô tả bài toán

Giá vàng phụ thuộc vào nhiều yếu tố như tỷ giá USD, lạm phát và giá dầu. Trong bài này, chúng ta sẽ xây dựng mô hình **Hồi quy tuyến tính** để dự đoán giá vàng dựa trên ba yếu tố:

- **USD Index:** Giá trị chỉ số USD.
- **Lạm phát (%):** Tỷ lệ lạm phát theo tháng.
- **Giá dầu (USD/thùng):** Giá dầu thô WTI.

Dữ liệu chứa 20 mẫu thu thập được, và chúng ta cần **huấn luyện mô hình hồi quy tuyến tính** bằng **Gradient Descent** để tối ưu tham số.

## 2 Dữ liệu

Dữ liệu giả lập gồm **20 mẫu**, với 3 đặc trưng đầu vào và 1 giá trị đầu ra (**giá vàng tính theo USD/ounce**):

## 3 Hướng dẫn giải

Chúng ta sử dụng mô hình **Hồi quy tuyến tính**:

$$y = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + b \quad (1)$$

Trong đó:

- $y$  là giá vàng (biến mục tiêu).
- $x_1, x_2, x_3$  lần lượt là USD Index, lạm phát và giá dầu.
- $w_0$  là bias (hệ số tự do).
- $w_1, w_2, w_3$  là các hệ số hồi quy.

Hàm mất mát Mean Squared Error (MSE):

$$J(w, b) = \frac{1}{2m} \sum_{i=1}^m (y_i - \hat{y}_i)^2 \quad (2)$$

USD Index	Lạm phát (%)	Giá dầu (USD)	Giá vàng (USD)
92.5	2.1	65.3	1800
93.2	2.5	67.2	1825
91.8	2.3	64.0	1795
94.0	2.8	70.1	1850
95.2	3.0	72.5	1880
96.1	3.2	74.3	1905
90.5	1.8	61.0	1750
92.0	2.0	63.2	1780
89.5	1.5	59.8	1725
97.0	3.5	76.2	1925
95.8	3.1	73.8	1890
94.5	2.9	71.5	1860
91.2	2.2	62.8	1775
90.0	1.7	60.5	1740
98.0	3.7	78.0	1950
99.2	4.0	80.5	1980
88.5	1.3	58.0	1700
87.8	1.1	56.5	1680
86.5	1.0	55.0	1650
100.0	4.2	82.0	2000

Bảng 1: Dữ liệu mẫu

Gradient Descent cập nhật tham số:

$$w_j := w_j - \alpha \cdot \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i) x_{i,j} \quad (3)$$

$$b := b - \alpha \cdot \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i) \quad (4)$$

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # Dữ liệu mẫu
5 X = np.array([...])  # Như trong bảng dữ liệu
6 y = np.array([...])
7
8 class LinearRegression:
9     def __init__(self):
10         # Khởi tạo weights và bias bằng 0
11         self.w = np.zeros(X.shape[1])  # weights cho 3 features
12         self.b = 0  # bias
13
14     def compute_gradients(self, X, y):
15         # Your code here #

```

```
16         return dw, db
17
18
19     def fit(self, X, y, learning_rate=0.001, epochs=10):
20         for epoch in range(epochs):
21             # Your code here #
22             print(f'Epoch {epoch}, Loss: {loss:.2f}')
23
24     def predict(self, X):
25         # Your code here #
26
27 # Normalize chi features X de trinh overflow
28 X_normalized = (X - np.mean(X, axis=0)) / np.std(X, axis=0)
29
30 # Khởi tạo và huấn luyện mô hình
31 model = LinearRegression()
32 model.fit(X_normalized, y, learning_rate=0.05, epochs=1000)
33
34 # In ra weights và bias cuối cùng
35 print("\nFinal weights:", model.w)
36 print("Final bias:", model.b)
37
38 # Dự đoán và so sánh kết quả
39 y_pred = model.predict(X_normalized)
40
41 print("\nSo sánh kết quả thực tế và dự đoán:")
42 for i in range(5):
43     print(f"Thực tế: {y[i]:.2f}, Dự đoán: {y_pred[i]:.2f}")
44
45 ######
46 # Đáp số
47 # So sánh kết quả thực tế và dự đoán:
48 # Thực tế: 1800.00, Dự đoán: 1794.78
49 # Thực tế: 1825.00, Dự đoán: 1822.10
50 # Thực tế: 1795.00, Dự đoán: 1789.47
51 # Thực tế: 1850.00, Dự đoán: 1850.66
52 # Thực tế: 1880.00, Dự đoán: 1878.38
53 ######
```

# Basic Python - Work with MySQL Database

*Hoàng-Nguyễn Vũ*

## 1 Lý thuyết

### 1.1 Cách tạo bảng trong SQL

Bảng trong SQL giúp lưu trữ dữ liệu dưới dạng hàng và cột. Khi tạo bảng, cần định nghĩa các cột, kiểu dữ liệu và ràng buộc.

Cú pháp tạo bảng:

```
1 CREATE TABLE table_name (
2     column1 datatype constraints ,
3     column2 datatype constraints ,
4     ...
5 );
```

Ví dụ:

```
1 CREATE TABLE Employees (
2     EmployeeID INT PRIMARY KEY ,
3     FullName VARCHAR(100) NOT NULL ,
4     Email VARCHAR(100) UNIQUE ,
5     DepartmentID INT ,
6     Salary DECIMAL(10 , 2)
7 );
```

### 1.2 Chuẩn hóa trong SQL

#### Chuẩn 1NF (First Normal Form)

- Mỗi cột chỉ chứa một giá trị duy nhất.
- Không có nhóm dữ liệu lặp lại.

#### Chuẩn 2NF (Second Normal Form)

- Đã đạt 1NF.
- Mọi thuộc tính không khóa phải phụ thuộc hoàn toàn vào khóa chính.

#### Chuẩn 3NF (Third Normal Form)

- Đã đạt 2NF.
- Không có phụ thuộc bắc cầu giữa các thuộc tính không khóa.

### 1.3 Các câu lệnh SQL cơ bản

Lệnh SELECT:

```
1 SELECT column1, column2 FROM table_name WHERE condition;
```

Lệnh INSERT:

```
1 INSERT INTO table_name (column1, column2) VALUES (value1, value2);
```

Lệnh UPDATE:

```
1 UPDATE table_name SET column1 = value1 WHERE condition;
```

Lệnh DELETE:

```
1 DELETE FROM table_name WHERE condition;
```

## 2 Bài tập thực hành SQL: Hệ thống Quản lý Doanh Số

### 2.1 Bối cảnh bài toán

Một công ty muốn xây dựng hệ thống quản lý doanh số bán hàng, theo dõi thông tin nhân viên bán hàng, đơn hàng và khách hàng.

### 2.2 Cấu trúc bảng dữ liệu

Bảng Salespersons (Nhân viên bán hàng)

- SalespersonID (INT, khóa chính)
- FullName (VARCHAR, tên đầy đủ)
- Email (VARCHAR, email duy nhất)
- PhoneNumber (VARCHAR, số điện thoại)

Bảng Customers (Khách hàng)

- CustomerID (INT, khóa chính)
- FullName (VARCHAR, tên khách hàng)
- Email (VARCHAR, email duy nhất)
- PhoneNumber (VARCHAR, số điện thoại)

Bảng Orders (Đơn hàng)

- OrderID (INT, khóa chính)
- CustomerID (INT, khóa ngoại)

- SalespersonID (INT, khóa ngoại)
- OrderDate (DATE, ngày đặt hàng)
- TotalAmount (DECIMAL, tổng giá trị đơn hàng)

### 2.3 Bài tập

- Câu 1: Viết lệnh SQL để tạo bảng Salespersons.
- Câu 2: Viết lệnh SQL để tạo bảng Customers.
- Câu 3: Viết lệnh SQL để tạo bảng Orders.
- Câu 4: Chèn dữ liệu vào bảng Salespersons.
- Câu 5: Chèn dữ liệu vào bảng Customers.
- Câu 6: Chèn dữ liệu vào bảng Orders.
- Câu 7: Liệt kê tất cả các đơn hàng của nhân viên có SalespersonID = 1.
- Câu 8: Tìm khách hàng có số lượng đơn hàng nhiều nhất.
- Câu 9: Tính tổng doanh số của từng nhân viên.
- Câu 10: Tìm nhân viên có doanh số cao nhất.
- Câu 11: Cập nhật tổng giá trị đơn hàng có OrderID = 2 thành 4.500.000
- Câu 12: Cập nhật số điện thoại của khách hàng có CustomerID = 3
- Câu 13: Xóa tất cả các đơn hàng của khách hàng có CustomerID = 2

# SQL Nâng Cao: Subqueries và Temporary Tables

*Hoàng-Nguyễn Vũ*

## 1 Lý thuyết

### 1.1 Subqueries (Câu truy vấn lồng)

#### Định nghĩa

Subquery là một truy vấn SQL được đặt bên trong một truy vấn khác. Truy vấn con được thực thi trước và kết quả của nó được sử dụng trong truy vấn cha.

#### Các loại Subquery

1. **Scalar Subquery:** Trả về đúng một giá trị (1 dòng, 1 cột)
2. **Row Subquery:** Trả về một dòng có nhiều cột
3. **Table Subquery:** Trả về nhiều dòng và nhiều cột, thường sử dụng trong FROM

#### Vị trí sử dụng phổ biến

- Trong SELECT để tính toán động
- Trong WHERE hoặc HAVING để lọc dữ liệu
- Trong FROM như một bảng phụ (table subquery)
- Trong EXISTS, IN, ANY, ALL để so sánh tập dữ liệu

#### Ưu điểm

- Cho phép viết truy vấn một cách trực quan, dễ hiểu
- Giúp tái sử dụng logic xử lý phức tạp mà không cần viết nhiều truy vấn riêng biệt

#### Ví dụ minh họa

```
1 SELECT name
2 FROM customers
3 WHERE id IN (
4     SELECT customer_id
5     FROM orders
6     WHERE total > 1000
7 );
```

```
1 SELECT name ,  
2      (SELECT SUM(total)  
3       FROM orders  
4      WHERE orders.customer_id = customers.id) AS total_spent  
5 FROM customers;
```

```
1 SELECT *  
2 FROM (  
3   SELECT customer_id, SUM(total) AS total_spent  
4   FROM orders  
5   GROUP BY customer_id  
6 ) AS spending  
7 WHERE total_spent > 1000;
```

```
1 SELECT name  
2 FROM customers c  
3 WHERE EXISTS (  
4   SELECT 1  
5   FROM orders o  
6   WHERE o.customer_id = c.id AND o.total > 1000  
7 );
```

## 1.2 Temporary Tables (Bảng tạm thời)

### Định nghĩa

Bảng tạm là bảng tồn tại trong thời gian phiên làm việc hiện tại (session). Sau khi kết thúc session hoặc đóng kết nối, bảng sẽ bị xóa tự động (tùy vào hệ quản trị cơ sở dữ liệu).

### Đặc điểm

- Không tồn tại lâu dài trong cơ sở dữ liệu
- Dữ liệu được lưu trên ổ đĩa hoặc bộ nhớ RAM tùy vào hệ quản trị
- Có thể truy vấn, cập nhật, và xóa như bảng thường
- Thường được dùng để xử lý dữ liệu tạm trong ETL hoặc khi cần chia bước xử lý phức tạp

## Cú pháp tạo bảng tạm

```
1 CREATE TEMPORARY TABLE temp_table_name AS
2 SELECT ...
3 FROM ...
4 WHERE ...;
```

## Ưu điểm của Temporary Table

- Tăng hiệu suất khi xử lý dữ liệu nhiều bước
- Giảm độ phức tạp của truy vấn lồng
- Có thể tái sử dụng cho nhiều truy vấn trong cùng session

## Ví dụ sử dụng TEMP TABLE

```
1 -- Bước 1: Tạo bảng tạm top_customers
2 CREATE TEMPORARY TABLE top_customers AS
3 SELECT customer_id, SUM(total) AS total_spent
4 FROM orders
5 GROUP BY customer_id
6 HAVING SUM(total) > 1000;
7
8 -- Bước 2: Truy vấn từ bảng tạm
9 SELECT c.name, t.total_spent
10 FROM customers c
11 JOIN top_customers t ON c.id = t.customer_id;
12
13 -- (Tùy chọn) Xoá bảng tạm nếu muốn
14 DROP TABLE top_customers;
```

## Lưu ý

- Trong một số hệ quản trị như MySQL, bạn cần có quyền để tạo bảng tạm
- Với PostgreSQL, TEMP TABLE bị xóa sau khi session kết thúc
- TEMP TABLE không nên dùng khi cần lưu dữ liệu lâu dài hoặc chia sẻ giữa nhiều phiên

## 2 Dữ liệu mẫu

Bảng customers

<b>id</b>	<b>name</b>	<b>city</b>
1	Alice	Hanoi
2	Bob	Ho Chi Minh
3	Charlie	Hanoi

Bảng orders

<b>id</b>	<b>customer_id</b>	<b>order_date</b>	<b>total</b>
1	1	2024-01-10	500
2	1	2024-03-01	800
3	2	2024-02-20	1200
4	3	2024-01-15	200

Bảng products

<b>id</b>	<b>name</b>	<b>price</b>
1	Laptop	1500
2	Mouse	50
3	Keyboard	100
4	Monitor	300

Bảng order\_items

<b>id</b>	<b>order_id</b>	<b>product_id</b>	<b>quantity</b>
1	1	2	2
2	1	3	1
3	2	1	1
4	2	4	1
5	3	1	1

Bảng employees

<b>id</b>	<b>name</b>	<b>department</b>
1	David	Sales
2	Emma	Support
3	Frank	Sales

## Bảng order\_assignments

<b>id</b>	<b>order_id</b>	<b>employee_id</b>
1	1	1
2	2	3
3	3	2

## Bài tập: Subqueries và Temporary Tables

### 1. Subquery trong WHERE

Tìm tất cả khách hàng đã từng mua đơn hàng chứa sản phẩm có giá lớn hơn 1000.

### 2. Subquery trong SELECT

Liệt kê tên từng khách hàng và tổng giá trị tất cả đơn hàng mà họ đã đặt.

### 3. Subquery trong FROM

Hiển thị **top 2** khách hàng có tổng chi tiêu cao nhất (dựa trên cột **total** của đơn hàng).

### 4. Subquery kết hợp EXISTS

Tìm tên nhân viên thuộc phòng Sales đã từng xử lý đơn hàng có chứa sản phẩm tên "Monitor".

### 5. Subquery trong HAVING

Tìm các khách hàng có tổng chi tiêu cao hơn **mức trung bình** của tất cả khách hàng.

### 6. Tạo bảng tạm khách hàng VIP

Tạo bảng tạm **vip\_customers** gồm những khách hàng có tổng giá trị đơn hàng lớn hơn 1000.

Sau đó, hiển thị tên và tổng chi tiêu của các khách hàng trong bảng tạm này.

### 7. Tạo bảng tạm sản phẩm phổ biến

Tạo bảng tạm **popular\_products** gồm những sản phẩm được bán ra với tổng số lượng từ 2 trở lên.

Hiển thị tên và tổng số lượng bán ra của các sản phẩm này.

### 8. Subquery lồng nhau nhiều cấp

Liệt kê tên các khách hàng có đơn hàng được xử lý bởi nhân viên có tổng doanh số lớn hơn 1000.

### 9. Kết hợp Temp Table và Subquery

- a. Tạo bảng tạm **high\_value\_orders** gồm những đơn hàng có tổng tiền > 800.

- b. Hiển thị tên khách hàng của các đơn hàng đó.
10. **Subquery tính tổng giá trị đơn hàng**  
Tính tổng giá trị thực tế của từng đơn hàng bằng cách lấy `price × quantity` theo từng dòng sản phẩm.

# Mini Project: Hệ thống Quản lý Bán Hàng với MySQL

*Hoàng-Nguyễn Vũ*

## 1 Mô tả bài toán

Một công ty cần xây dựng hệ thống quản lý bán hàng để theo dõi:

- **Sản phẩm:** Danh sách sản phẩm, giá bán, số lượng tồn kho.
- **Khách hàng:** Thông tin khách hàng để theo dõi lịch sử mua hàng.
- **Nhân viên bán hàng:** Ghi nhận ai thực hiện giao dịch.
- **Đơn hàng:** Mỗi đơn hàng cần ghi nhận khách hàng, nhân viên bán hàng, tổng giá trị đơn hàng.
- **Chi tiết đơn hàng:** Ghi nhận từng sản phẩm trong đơn hàng.

Yêu cầu:

- Phân tích quan hệ giữa các bảng.
- Xác định kiểu dữ liệu phù hợp.
- Viết lệnh SQL để tạo, chèn, cập nhật và thống kê dữ liệu.

## 2 Mô tả chi tiết các bảng dữ liệu

### 2.1 Bảng Products (Sản phẩm)

- **ProductID** (INT, PRIMARY KEY) - Mã định danh duy nhất cho mỗi sản phẩm.
- **ProductName** (VARCHAR(100)) - Tên sản phẩm.
- **Category** (VARCHAR(50)) - Danh mục sản phẩm (Laptop, Điện thoại, Phụ kiện,...).
- **Price** (DECIMAL(10,2)) - Giá bán sản phẩm.
- **StockQuantity** (INT) - Số lượng sản phẩm tồn kho.

### 2.2 Bảng Customers (Khách hàng)

- **CustomerID** (INT, PRIMARY KEY) - Mã định danh khách hàng.
- **FullName** (VARCHAR(100)) - Họ và tên đầy đủ của khách hàng.
- **PhoneNumber** (VARCHAR(15), UNIQUE) - Số điện thoại khách hàng.
- **Email** (VARCHAR(100), UNIQUE) - Địa chỉ email khách hàng.

### 2.3 Bảng Salespersons (Nhân viên bán hàng)

- **SalespersonID** (INT, PRIMARY KEY) - Mã định danh nhân viên bán hàng.
- **FullName** (VARCHAR(100)) - Họ và tên nhân viên.
- **PhoneNumber** (VARCHAR(15), UNIQUE) - Số điện thoại nhân viên bán hàng.

### 2.4 Bảng Orders (Đơn hàng)

- **OrderID** (INT, PRIMARY KEY) - Mã định danh đơn hàng.
- **CustomerID** (INT, FOREIGN KEY) - Mã khách hàng (liên kết với Customers).
- **SalespersonID** (INT, FOREIGN KEY) - Mã nhân viên bán hàng (liên kết với Salespersons).
- **OrderDate** (DATE) - Ngày đặt hàng.
- **TotalAmount** (DECIMAL(10,2)) - Tổng giá trị đơn hàng.

### 2.5 Bảng OrderDetails (Chi tiết đơn hàng)

- **OrderDetailID** (INT, PRIMARY KEY) - Mã định danh chi tiết đơn hàng.
- **OrderID** (INT, FOREIGN KEY) - Mã đơn hàng (liên kết với Orders).
- **ProductID** (INT, FOREIGN KEY) - Mã sản phẩm (liên kết với Products).
- **Quantity** (INT) - Số lượng sản phẩm mua trong đơn hàng.
- **Subtotal** (DECIMAL(10,2)) - Tổng tiền của sản phẩm trong đơn hàng (**Quantity \* Price**).

## 3 Bảng dữ liệu mẫu

### 3.1 Bảng Products (Sản phẩm)

ProductID	ProductName	Category	Price	StockQuantity
1	Laptop Dell XPS	Laptop	25000000	10
2	iPhone 14	Điện thoại	22000000	15
3	Tai nghe Sony	Phụ kiện	3000000	20
4	Bàn phím Cơ	Phụ kiện	1500000	50
5	MacBook Air M2	Laptop	28000000	8

### 3.2 Bảng Customers (Khách hàng)

CustomerID	FullName	PhoneNumber	Email
1	Nguyễn Văn A	0987654321	a@example.com
2	Trần Thị B	0976543210	b@example.com
3	Lê Văn C	0965432109	c@example.com
4	Phạm Thị D	0945678123	d@example.com
5	Bùi Văn E	0936789123	e@example.com

### 3.3 Bảng Salespersons (Nhân viên bán hàng)

SalespersonID	FullName	PhoneNumber
1	Lê Hoàng Nam	0934567890
2	Trần Quốc Bảo	0945678123
3	Nguyễn Minh Hà	0956789123

### 3.4 Bảng Orders (Đơn hàng)

OrderID	CustomerID	SalespersonID	OrderDate	TotalAmount
1	1	1	2024-03-10	25000000
2	2	2	2024-03-11	3000000
3	3	3	2024-03-12	22000000

### 3.5 Bảng OrderDetails (Chi tiết đơn hàng)

OrderDetailID	OrderID	ProductID	Quantity	Subtotal
1	1	1	1	25000000
2	2	3	1	3000000
3	3	2	1	22000000

## 4 Bài tập thực hành SQL

### 4.1 Bước 1: Tạo bảng

- Viết lệnh SQL để tạo bảng Products.
- Viết lệnh SQL để tạo bảng Customers.
- Viết lệnh SQL để tạo bảng Salespersons.
- Viết lệnh SQL để tạo bảng Orders.
- Viết lệnh SQL để tạo bảng OrderDetails.

## 4.2 Bước 2: Chèn dữ liệu

1. Thêm ít nhất 5 sản phẩm vào bảng Products.
2. Thêm ít nhất 5 khách hàng vào bảng Customers.
3. Thêm ít nhất 3 nhân viên vào bảng Salespersons.
4. Thêm ít nhất 5 đơn hàng vào bảng Orders.
5. Thêm chi tiết sản phẩm vào bảng OrderDetails.

## 4.3 Bước 3: Truy vấn dữ liệu

1. Truy vấn danh sách tất cả sản phẩm còn hàng.
2. Truy vấn danh sách đơn hàng của khách hàng có CustomerID = 1.
3. Truy vấn tổng doanh số của từng nhân viên.

## 4.4 Bước 4: Thống kê nâng cao

1. Nhân viên bán hàng có tổng doanh số cao nhất.
2. Tính tổng doanh thu theo từng danh mục sản phẩm.
3. Tìm top 3 khách hàng có tổng chi tiêu cao nhất.
4. Tìm tháng có doanh thu cao nhất.
5. Tìm sản phẩm bán chạy nhất theo số lượng.

# SQL Nâng Cao: Stored Procedure

*Hoàng-Nguyễn Vũ*

## 1 Lý thuyết

### 1.1. Định nghĩa

Stored Procedure (Thủ tục lưu trữ) là một tập hợp các câu lệnh SQL được lưu trữ trong cơ sở dữ liệu và có thể được gọi để thực thi nhiều lần.

**Lợi ích:**

- Giúp tái sử dụng logic nghiệp vụ
- Giảm độ trễ do ít giao tiếp với client
- Có thể chứa logic rẽ nhánh, kiểm tra điều kiện

### 1.2. Cú pháp (MySQL)

```
1 DELIMITER //
2
3 CREATE PROCEDURE procedure_name(
4     IN input_param INT,
5     OUT output_param VARCHAR(100)
6 )
7 BEGIN
8     -- Câu lệnh SQL ở đây
9 END //
10
11 DELIMITER ;
```

### 1.3. Gọi procedure

```
1 CALL procedure_name(input_value, @output_var);
2 SELECT @output_var;
```

### 1.4. Ví dụ đơn giản

Tạo procedure tính tổng chi tiêu của 1 khách hàng:

```
1 DELIMITER //
2
3 CREATE PROCEDURE GetCustomerSpending(
4     IN customerId INT,
5     OUT total_spent INT
6 )
```

```
7 BEGIN
8   SELECT SUM(total)
9   INTO total_spent
10  FROM orders
11  WHERE customer_id = customerId;
12 END //
13
14 DELIMITER ;
15
16 -- Gọi:
17 CALL GetCustomerSpending(1, @spending);
18 SELECT @spending;
```

## 2 Dữ liệu mẫu

Bảng customers

id	name	city
1	Alice	Hanoi
2	Bob	Ho Chi Minh
3	Charlie	Hanoi

Bảng orders

id	customer_id	order_date	total
1	1	2024-01-10	500
2	1	2024-03-01	800
3	2	2024-02-20	1200
4	3	2024-01-15	200

Bảng products

id	name	price
1	Laptop	1500
2	Mouse	50
3	Keyboard	100
4	Monitor	300

## Bảng order\_items

id	order_id	product_id	quantity
1	1	2	2
2	1	3	1
3	2	1	1
4	2	4	1
5	3	1	1

## Bảng employees

id	name	department
1	David	Sales
2	Emma	Support
3	Frank	Sales

## Bảng order\_assignments

id	order_id	employee_id
1	1	1
2	2	3
3	3	2

## Bài tập Stored Procedure

1. **Tạo procedure lấy tổng số đơn hàng của 1 khách hàng**  
 Đầu vào: customer\_id (INT)  
 Đầu ra: order\_count (INT)
2. **Tạo procedure lấy tên nhân viên theo mã ID**  
 Đầu vào: employee\_id (INT)  
 Đầu ra: employee\_name (VARCHAR)
3. **Tạo procedure cập nhật thành phố của khách hàng**  
 Đầu vào: customer\_id (INT), new\_city (VARCHAR)
4. **Tạo procedure in ra các sản phẩm có giá từ X trở lên**  
 Đầu vào: min\_price (INT)
5. **Tạo procedure xoá đơn hàng có tổng < X**  
 Đầu vào: min\_total (INT)

# Word Cloud: Khái niệm và Ứng dụng trong Xử lý Ngôn ngữ Tự nhiên

*Hoàng-Nguyễn Vũ*

## 1 Lý thuyết

WordCloud (Đám mây từ) là một kỹ thuật trực quan hóa văn bản, trong đó kích thước của từ phản ánh tần suất xuất hiện của nó trong tập dữ liệu. Đây là một công cụ đơn giản nhưng hiệu quả để khám phá dữ liệu văn bản ban đầu, giúp nhận diện nhanh các từ khóa chính trong tài liệu hoặc tập hợp văn bản lớn.

## Ứng dụng

- Phân tích phản hồi khách hàng, khảo sát, đánh giá.
- Phân tích nội dung truyền thông xã hội.
- Trực quan hóa dữ liệu văn bản trong giảng dạy và nghiên cứu.

## Ưu điểm

- Trực quan, dễ hiểu, dễ tạo bằng các thư viện Python.
- Có thể tùy chỉnh về hình dạng, màu sắc, phông chữ.

## Hạn chế

- Không phản ánh được ngữ cảnh sử dụng từ.
- Dễ gây hiểu sai nếu chưa xử lý tốt các từ dừng (stopwords).

## Các bước tạo WordCloud

1. Thu thập và đọc dữ liệu văn bản.
2. Tiền xử lý văn bản: loại bỏ ký tự đặc biệt, chuyển về chữ thường, xóa từ dừng.
3. Kết hợp văn bản và tạo WordCloud bằng thư viện.
4. Hiển thị kết quả bằng công cụ vẽ (matplotlib).

## 2 Ví dụ và Code mẫu

Ví dụ sau đây minh họa cách tạo WordCloud từ một file CSV chứa cột văn bản.

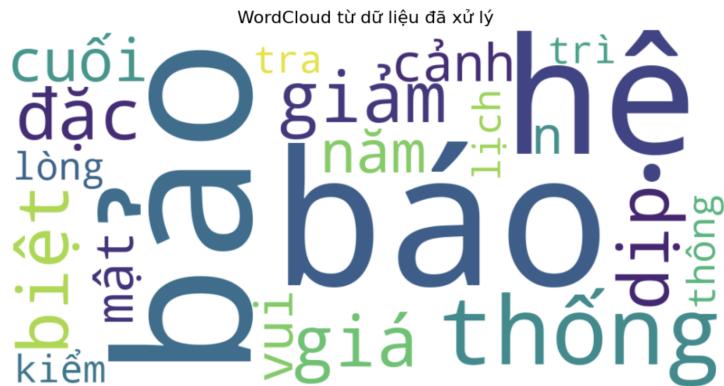
## Chuẩn bị dữ liệu

Tạo File data.csv có cấu trúc như sau:

text  
Subject: Giám giá đặc biệt dịp cuối năm!  
Cảnh báo bảo mật hệ thống \n Vui lòng kiểm tra.  
Thông báo: Lịch bảo trì hệ thống

## Code Python mẫu

## Kết quả:



## Ghi chú

- Thư viện `wordcloud` cần được cài đặt trước bằng `pip install wordcloud`.
  - Có thể sử dụng ảnh mask để tạo hình dạng tùy chỉnh cho WordCloud (trái tim, ngôi sao, logo...).

### 3 Bài tập

Cho dữ liệu Game of Thrones jon snow data.csv, thực hiện các yêu cầu sau:

1. **Đọc dữ liệu và thực hiện chuẩn hóa** (loại bỏ phần văn bản subject trước dấu “:”, và các ký tự đặc biệt như “\r”, “\n”).
  2. **Tạo biểu đồ WordCloud** có kết quả gợi ý như sau:



3. Cho tập tin hình ảnh jon-snow.jpg, hãy tạo biểu đồ WordCloud có kết quả gợi ý như hình sau:



# SQL Nâng Cao: Trigger

*Hoàng-Nguyễn Vũ*

## 1 Lý thuyết

### 1.1. Định nghĩa

**Trigger** (bẫy) là một đoạn mã SQL được thực thi **tự động** khi có sự kiện xảy ra trên một bảng như INSERT, UPDATE hoặc DELETE.

Công dụng:

- Ghi log thay đổi dữ liệu
- Kiểm tra tính hợp lệ (validation logic)
- Đồng bộ dữ liệu giữa các bảng
- Kiểm soát bảo mật dữ liệu

### 1.2. Cú pháp Trigger (MySQL)

```
1 CREATE TRIGGER trigger_name
2 AFTER INSERT ON table_name
3 FOR EACH ROW
4 BEGIN
5     -- Câu lệnh SQL tự động chạy
6 END;
```

Các loại thời điểm kích hoạt trigger:

- BEFORE INSERT / UPDATE / DELETE
- AFTER INSERT / UPDATE / DELETE

Các từ khóa:

- NEW.column – giá trị mới được INSERT hoặc UPDATE
- OLD.column – giá trị cũ bị UPDATE hoặc DELETE

### 1.3. Ví dụ: Ghi log khi INSERT vào bảng orders

```
1 CREATE TABLE order_logs (
2     id INT AUTO_INCREMENT PRIMARY KEY,
3     log_message TEXT,
4     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
5 );
6
```

```

7 CREATE TRIGGER log_new_order
8 AFTER INSERT ON orders
9 FOR EACH ROW
10 BEGIN
11     INSERT INTO order_logs(log_message)
12     VALUES (CONCAT('New order ID: ', NEW.id, ' - total: ', NEW.total));
13 END;

```

## 2 Dữ liệu mẫu

Bảng customers

id	name	city
1	Alice	Hanoi
2	Bob	Ho Chi Minh
3	Charlie	Hanoi

Bảng orders

id	customer_id	order_date	total
1	1	2024-01-10	500
2	1	2024-03-01	800
3	2	2024-02-20	1200
4	3	2024-01-15	200

Bảng products

id	name	price
1	Laptop	1500
2	Mouse	50
3	Keyboard	100
4	Monitor	300

Bảng order\_items

id	order_id	product_id	quantity
1	1	2	2
2	1	3	1
3	2	1	1
4	2	4	1
5	3	1	1

## Bảng employees

<b>id</b>	<b>name</b>	<b>department</b>
1	David	Sales
2	Emma	Support
3	Frank	Sales

## Bảng order\_assignments

<b>id</b>	<b>order_id</b>	<b>employee_id</b>
1	1	1
2	2	3
3	3	2

## Bài tập

### 1. Ghi log khi xoá đơn hàng

Tạo trigger để ghi log vào bảng `order_logs` mỗi khi đơn hàng bị xoá khỏi bảng `orders`.

### 2. Tự động cập nhật tổng tiền đơn hàng

Tạo trigger trên bảng `order_items` để tự động cập nhật tổng tiền của đơn hàng trong bảng `orders` khi thêm dòng sản phẩm mới.

### 3. Ngăn không cho chỉnh sửa giá sản phẩm nếu lớn hơn 1000

Tạo trigger BEFORE UPDATE trên bảng `products` để huỷ cập nhật nếu giá mới vượt quá 1000.

### 4. Tạo bản sao lưu khi xoá khách hàng

Mỗi khi xoá khách hàng, ghi lại thông tin vào bảng `customers_deleted` để lưu trữ.

### 5. Cảnh báo khi khách hàng từ thành phố “Hanoi” đặt đơn hàng lớn hơn 1000

Ghi log vào bảng `order_alerts` khi khách hàng từ “Hanoi” đặt đơn hàng có tổng lớn hơn 1000.

# Phân tích IQR và Phát hiện Outlier trong Dữ liệu Thống kê

Hoàng-Nguyễn Vũ

## 1 Lý thuyết

Trong phân tích thống kê, **IQR (Interquartile Range)** hay *khoảng tứ phân vị*, là một thước đo quan trọng dùng để đánh giá **mức độ phân tán của dữ liệu**. IQR thể hiện phạm vi chứa **50% số liệu trung tâm** của một tập dữ liệu, được xác định bởi hiệu số giữa phần tư thứ ba (Q3) và phần tư thứ nhất (Q1):

$$\text{IQR} = Q3 - Q1$$

IQR có ưu điểm là **không bị ảnh hưởng bởi các giá trị cực đoan**, nên rất phù hợp để mô tả sự phân bố trung tâm của các tập dữ liệu không phân phối chuẩn.

## 2 Cách tính Quartile

### 2.1 Định nghĩa

**Quartile (Tứ phân vị)** là các giá trị chia tập dữ liệu đã được sắp xếp thành bốn phần bằng nhau, mỗi phần tương ứng với 25% số quan sát.

- **Q1 (Phần tư thứ nhất):** 25% số dữ liệu nhỏ hơn hoặc bằng giá trị này.
- **Q2 (Trung vị):** 50% số dữ liệu nhỏ hơn hoặc bằng giá trị này.
- **Q3 (Phần tư thứ ba):** 75% số dữ liệu nhỏ hơn hoặc bằng giá trị này.

### 2.2 Quy trình tính toán

Giả sử tập dữ liệu đã được sắp xếp tăng dần:

$$3, 5, 7, 8, 12, 13, 14, 18, 21$$

- $Q2 (\text{Trung vị}) = 12$
- $Q1 = \frac{5+7}{2} = 6$
- $Q3 = \frac{14+18}{2} = 16$
- $\text{IQR} = 16 - 6 = 10$

## 3 Outlier và Phát hiện Outlier bằng IQR

### 3.1 Định nghĩa Outlier

**Outlier (Giá trị ngoại lai)** là các điểm dữ liệu có giá trị khác biệt rõ rệt so với xu hướng chung của tập dữ liệu. Chúng có thể do lỗi nhập liệu hoặc là các hiện tượng đặc biệt hiếm gặp.

### 3.2 Phát hiện Outlier bằng IQR

Dựa vào IQR, các ngưỡng để xác định outlier được tính như sau:

$$\text{Ngưỡng dưới (Lower Bound)} = Q1 - 1.5 \times IQR$$

$$\text{Ngưỡng trên (Upper Bound)} = Q3 + 1.5 \times IQR$$

Nếu một giá trị nằm ngoài khoảng [Lower Bound, Upper Bound] thì được coi là **outlier**.

### 3.3 Ví dụ minh họa

Tập dữ liệu:

$$3, 5, 7, 8, 12, 13, 14, 18, 21, 100$$

- $Q1 = 6, Q3 = 16 \Rightarrow IQR = 10$
- Lower Bound =  $6 - 1.5 \times 10 = -9$
- Upper Bound =  $16 + 1.5 \times 10 = 31$
- Giá trị  $100 > 31 \Rightarrow$  là một outlier

## 4 Tại sao sử dụng hệ số 1.5?

Hệ số **1.5** trong phương pháp IQR được đề xuất bởi **John Tukey**, người phát triển biểu đồ hộp (boxplot). Lý do chọn hệ số này:

- Tương đương khoảng 2.7 độ lệch chuẩn trong phân phối chuẩn.
- Giúp cân bằng giữa việc phát hiện điểm bất thường thật sự và tránh loại bỏ các điểm hợp lệ.
- Hệ số nhỏ hơn (như 1.0) sẽ loại bỏ quá nhiều giá trị hợp lệ; hệ số lớn hơn (như 3.0) có thể bỏ sót outlier thật sự.

## 5 Kết luận

Phân tích IQR và phát hiện outlier là bước quan trọng trong tiền xử lý dữ liệu. Sử dụng ngưỡng  $1.5 \times \text{IQR}$  là một phương pháp đơn giản nhưng hiệu quả để xác định các giá trị bất thường mà không bị ảnh hưởng bởi dữ liệu cực đoan. Tuy nhiên, việc loại bỏ outlier cần được cân nhắc trong từng trường hợp cụ thể nhằm đảm bảo tính toàn vẹn và ý nghĩa của dữ liệu.

## 6 Bài tập

### Bài tập 1: Tính IQR từ danh sách số

**Đề bài:** Với danh sách sau:

```
1 data = [3, 7, 8, 5, 12, 14, 21, 13, 18]
```

- Sắp xếp danh sách.
- Tính Q1, Q3 và IQR.
- In kết quả ra màn hình.

### Bài tập 2: Phát hiện Outlier bằng IQR

**Đề bài:** Sử dụng danh sách:

```
1 data = [3, 5, 7, 8, 12, 13, 14, 18, 21, 100]
```

- Tính Q1, Q3 và IQR.
- Xác định ngưỡng dưới và trên.
- In ra các giá trị ngoại lai.

### Bài tập 3: Loại bỏ Outlier khỏi DataFrame

**Đề bài:**

```
1 import pandas as pd
2
3 df = pd.DataFrame({
4     'score': [55, 61, 70, 65, 68, 90, 91, 94, 300, 58]
5 })
```

- Tính IQR cho cột score.
- Tìm các giá trị outlier.
- Tạo DataFrame mới không chứa các outlier.

# SQL Nâng Cao: VIEW

*Hoàng-Nguyễn Vũ*

## 1 Lý thuyết

### 1.1. Định nghĩa

VIEW là bảng ảo được tạo ra từ một truy vấn SELECT. Dữ liệu trong VIEW không lưu trữ thật mà được lấy mỗi khi truy vấn VIEW.

**Lợi ích:**

- Giúp đơn giản hóa truy vấn phức tạp
- Bảo vệ dữ liệu nhạy cảm
- Dễ dàng tái sử dụng và bảo trì

### 1.2. Cú pháp tạo VIEW

```
1 CREATE VIEW view_name AS
2 SELECT column1, column2, ...
3 FROM table_name
4 WHERE condition;
```

### 1.3. Cập nhật VIEW

- Có thể UPDATE/INSERT/DELETE trên VIEW nếu nó dựa trên 1 bảng duy nhất và không chứa phép ‘JOIN’, ‘GROUP BY’, ‘DISTINCT’...

### 1.4. Xoá VIEW

```
1 DROP VIEW view_name;
```

### 1.5. Ví dụ:

Tạo VIEW hiển thị tên khách hàng và tổng chi tiêu:

```
1 CREATE VIEW customer_spending AS
2 SELECT c.name, SUM(o.total) AS total_spent
3 FROM customers c
4 JOIN orders o ON c.id = o.customer_id
5 GROUP BY c.name;
```

## 2 Dữ liệu mẫu

### Bảng customers

<b>id</b>	<b>name</b>	<b>city</b>
1	Alice	Hanoi
2	Bob	Ho Chi Minh
3	Charlie	Hanoi

### Bảng orders

<b>id</b>	<b>customer_id</b>	<b>order_date</b>	<b>total</b>
1	1	2024-01-10	500
2	1	2024-03-01	800
3	2	2024-02-20	1200
4	3	2024-01-15	200

### Bảng products

<b>id</b>	<b>name</b>	<b>price</b>
1	Laptop	1500
2	Mouse	50
3	Keyboard	100
4	Monitor	300

### Bảng order\_items

<b>id</b>	<b>order_id</b>	<b>product_id</b>	<b>quantity</b>
1	1	2	2
2	1	3	1
3	2	1	1
4	2	4	1
5	3	1	1

### Bảng employees

<b>id</b>	<b>name</b>	<b>department</b>
1	David	Sales
2	Emma	Support
3	Frank	Sales

## Bảng order\_assignments

id	order_id	employee_id
1	1	1
2	2	3
3	3	2

## Bài tập

1. Tạo VIEW hiển thị tên khách hàng và tổng tiền đã chi.
2. Tạo VIEW danh sách sản phẩm có giá trên 100.
3. Tạo VIEW hiển thị các đơn hàng có tổng tiền trên 1000, kèm tên khách hàng.
4. Tạo VIEW hiển thị thông tin nhân viên thuộc phòng "Sales".
5. Tạo VIEW cho bảng order\_items hiển thị thêm tổng tiền = price × quantity.
6. Xoá VIEW product\_above\_100 nếu tồn tại.
7. Tạo VIEW kết hợp nhiều bảng: khách hàng, đơn hàng và nhân viên xử lý đơn.
8. Tạo VIEW thống kê số lượng sản phẩm được bán theo từng loại sản phẩm.
9. Tạo VIEW chỉ cho phép xem các đơn hàng của khách hàng từ thành phố "Hanoi".
10. Tạo VIEW giúp kế toán xem đơn hàng gồm: order\_id, customer name, total.

# Tổng quan về Matplotlib, Seaborn và Plotly trong Trực quan hóa Dữ liệu

*Hoàng-Nguyễn Vũ*

## 1 Lý thuyết

Trong khoa học dữ liệu và trực quan hóa dữ liệu, ba thư viện phổ biến trong Python là:

- **Matplotlib**: Thư viện gốc cho vẽ đồ thị 2D trong Python.
- **Seaborn**: Một thư viện xây dựng trên Matplotlib, tối ưu cho thống kê.
- **Plotly**: Một thư viện hiện đại hỗ trợ trực quan hóa dữ liệu tương tác.

## 2 Matplotlib

### 2.1 Giới thiệu

Matplotlib là thư viện lâu đời nhất cho trực quan hóa dữ liệu trong Python, cho phép tạo các biểu đồ 2D tĩnh.

### 2.2 Ưu điểm

- Linh hoạt, hỗ trợ nhiều loại biểu đồ.
- Kiểm soát chi tiết từng thành phần của đồ thị.
- Tích hợp tốt với các công cụ khoa học dữ liệu như Pandas, NumPy.

### 2.3 Hạn chế

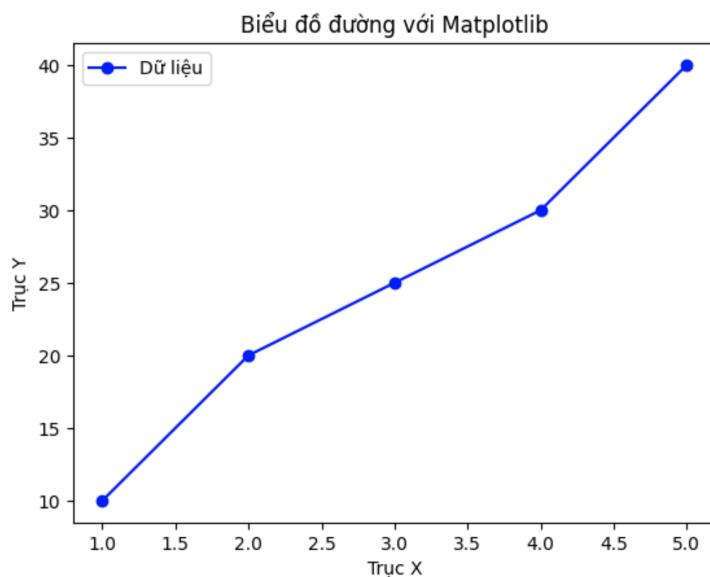
- Cú pháp phức tạp, cần nhiều dòng lệnh để tạo biểu đồ đẹp.
- Không hỗ trợ trực quan hóa tương tác.

### 2.4 Ví dụ Code Matplotlib

```
1 import matplotlib.pyplot as plt
2
3 x = [1, 2, 3, 4, 5]
4 y = [10, 20, 25, 30, 40]
5
6 plt.plot(x, y, marker='o', linestyle='--', color='b', label="Dữ liệu")
7 plt.xlabel("Trục X")
8 plt.ylabel("Trục Y")
```

```
9 plt.title("Biểu đồ đường với Matplotlib")
10 plt.legend()
11 plt.show()
```

Kết quả:



## 3 Seaborn

### 3.1 Giới thiệu

Seaborn là thư viện mở rộng của Matplotlib, giúp đơn giản hóa quá trình vẽ đồ thị thống kê với giao diện dễ sử dụng.

### 3.2 Ưu điểm

- Hỗ trợ các biểu đồ thống kê như phân bố, hồi quy.
- Giao diện đẹp mắt hơn Matplotlib mặc định.
- Tích hợp tốt với Pandas.

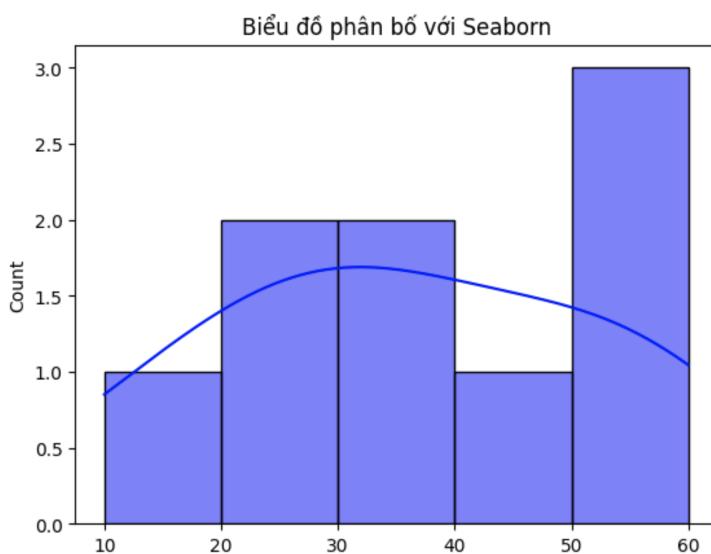
### 3.3 Hạn chế

- Không linh hoạt như Matplotlib.
- Không hỗ trợ tương tác.

### 3.4 Ví dụ Code Seaborn

```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3
4 # Dữ liệu giả lập
5 data = [10, 20, 25, 30, 40, 35, 50, 60, 55]
6
7 sns.histplot(data, kde=True, color='blue')
8 plt.title("Biểu đồ phân bố với Seaborn")
9 plt.show()
```

Kết quả:



## 4 Plotly

### 4.1 Giới thiệu

Plotly là thư viện mạnh mẽ hỗ trợ tạo biểu đồ **tương tác**, giúp người dùng dễ dàng khám phá dữ liệu hơn.

### 4.2 Ưu điểm

- Hỗ trợ tương tác mạnh mẽ (zoom, hover, lọc).
- Hỗ trợ 3D và trực quan hóa nâng cao.
- Dễ tích hợp với ứng dụng web (Dash).

### 4.3 Hạn chế

- Cài đặt nặng hơn so với Matplotlib và Seaborn.
- Không phù hợp cho các biểu đồ data có dung lượng lớn.

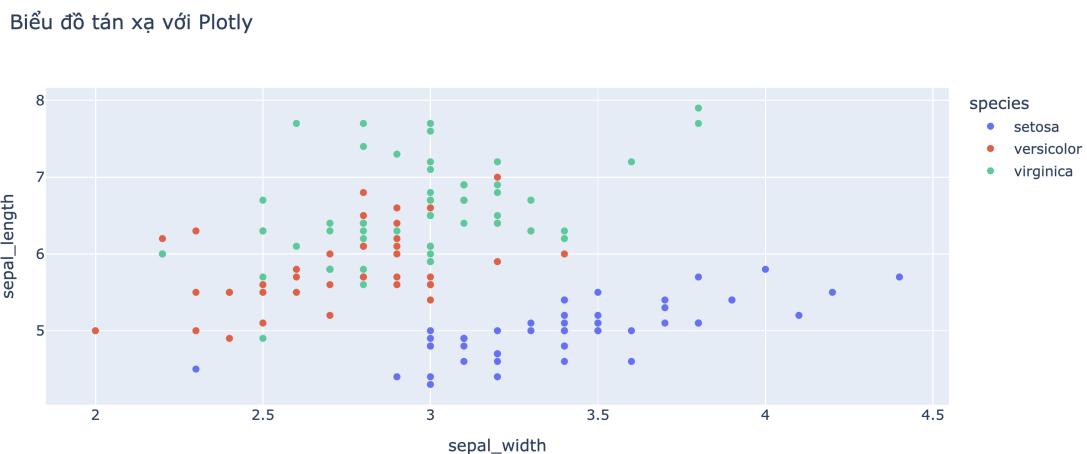
### 4.4 Ví dụ Code Plotly

```

1 import plotly.express as px
2
3 df = px.data.iris() # Dữ liệu mẫu
4
5 fig = px.scatter(df, x="sepal_width", y="sepal_length",
6                   color="species", title="Biểu đồ tán xạ với Plotly")
7 fig.show()

```

Kết quả:



## 5 So sánh Matplotlib, Seaborn và Plotly

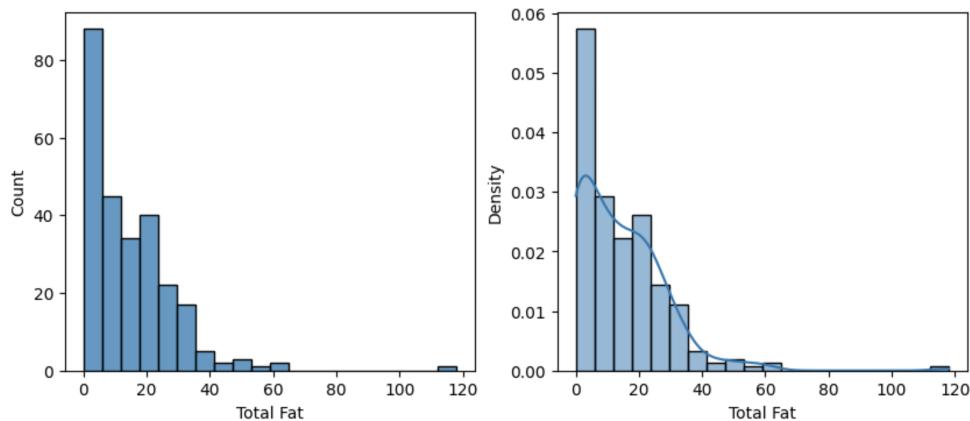
Tính năng	Matplotlib	Seaborn	Plotly
Loại đồ thị	2D	2D	2D/3D
Hỗ trợ tương tác	Không	Không	Có
Tích hợp với Pandas	Có	Có	Có
Biểu đồ thống kê	Cơ bản	Nâng cao	Nâng cao
Độ linh hoạt	Cao	Trung bình	Cao
Dễ sử dụng	Trung bình	Dễ	Trung bình

Bảng 1: So sánh Matplotlib, Seaborn và Plotly

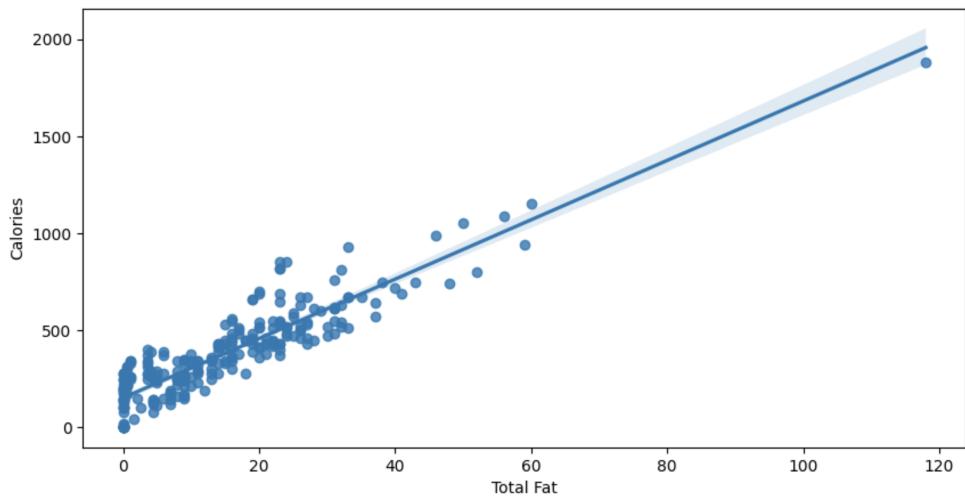
## 6 Bài tập

Cho dữ liệu **menu.csv**, thực hiện các yêu cầu sau:

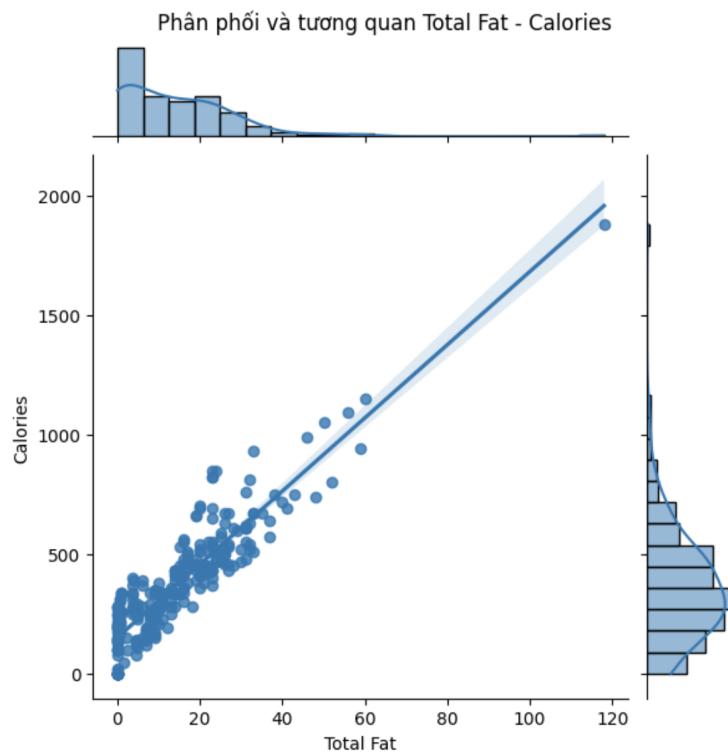
1. Đọc dữ liệu, hiển thị thông tin chung của dữ liệu : head, tail, info, describe.
2. Vẽ biểu đồ phân phối tần suất các món theo Total fat gợi ý như 2 hình sau:



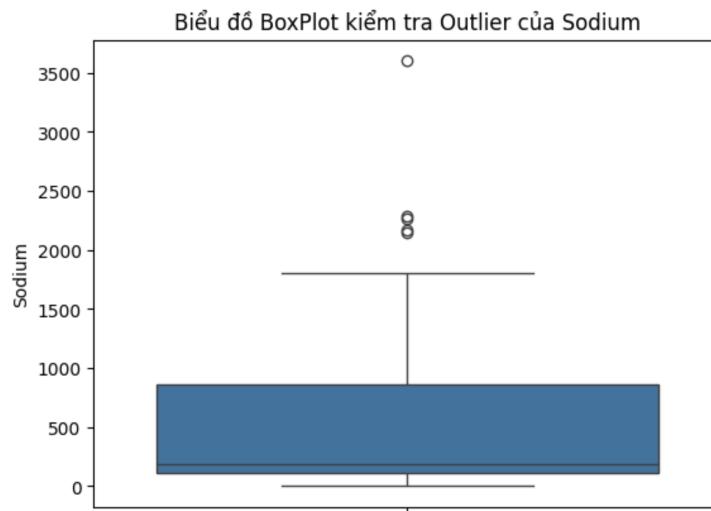
3. Nhận xét biểu đồ trên
4. Tính hệ số tương quan giữa 2 thuộc tính Total Fat và Calories, sau đó vẽ biểu đồ như hình sau



5. Vẽ biểu đồ thể hiện sự tương quan giữa Total Fat và Calories theo gợi ý như hình sau:

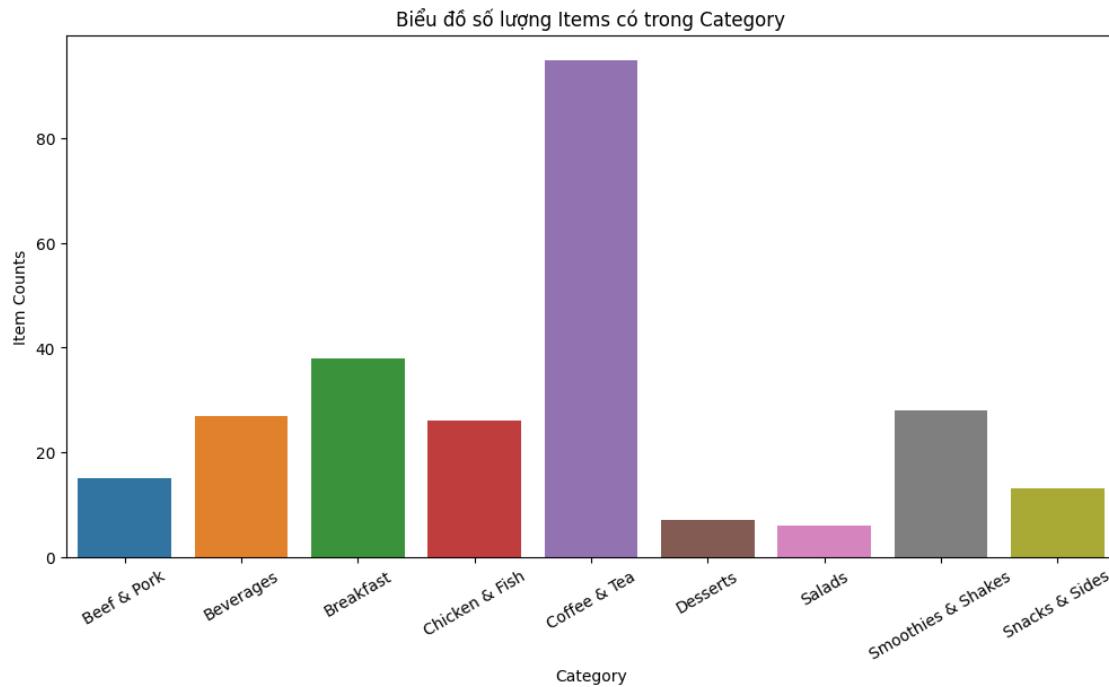


6. Vẽ biểu đồ kiểm tra dữ liệu của cột Sodium gợi ý như hình sau:



7. Dữ liệu của cột Sodium theo như hình trên có outliers hay không, nếu có thì loại bỏ tất cả các dòng trong data có outliers?

8. Nhóm dữ liệu Category và đếm theo Item, cho biết mỗi nhóm có bao nhiêu?
9. Vẽ biểu đồ thể hiện dữ liệu Category theo gợi ý như hình sau:



# SQL Nâng Cao: Index

*Hoàng-Nguyễn Vũ*

## 1 Lý thuyết

### 1.1. Định nghĩa

Index (chỉ mục) là một cấu trúc dữ liệu đặc biệt được sử dụng để tăng tốc độ truy vấn dữ liệu trong bảng SQL, tương tự như mục lục trong sách.

### 1.2. Lợi ích của Index

- Tăng hiệu suất truy vấn SELECT, WHERE, JOIN, ORDER BY
- Giảm thời gian tìm kiếm dữ liệu

### 1.3. Nhược điểm của Index

- Làm chậm INSERT, UPDATE, DELETE
- Tốn thêm bộ nhớ để lưu index

### 1.4. Cú pháp tạo Index

```
1 -- Index thông thường
2 CREATE INDEX index_name
3 ON table_name (column1, column2, ...);
4
5 -- Index duy nhất (UNIQUE)
6 CREATE UNIQUE INDEX unique_index_name
7 ON table_name (column_name);
```

### 1.5. Xoá Index

```
1 DROP INDEX index_name ON table_name;
```

### 1.6. Khi nào nên dùng Index?

- Cột thường dùng trong WHERE, JOIN, ORDER BY
- Cột là khoá chính hoặc khoá ngoại
- Cột có độ phân biệt cao (nhiều giá trị khác nhau)

## 1.7. Khi không nên dùng Index?

- Bảng nhỏ (ít bản ghi)
- Cột có ít giá trị khác nhau (low selectivity, ví dụ giới tính)
- Bảng thường xuyên ghi dữ liệu mới

## 2 Dữ liệu mẫu

Bảng customers

id	name	city
1	Alice	Hanoi
2	Bob	Ho Chi Minh
3	Charlie	Hanoi

Bảng orders

id	customer_id	order_date	total
1	1	2024-01-10	500
2	1	2024-03-01	800
3	2	2024-02-20	1200
4	3	2024-01-15	200

Bảng products

id	name	price
1	Laptop	1500
2	Mouse	50
3	Keyboard	100
4	Monitor	300

Bảng order\_items

id	order_id	product_id	quantity
1	1	2	2
2	1	3	1
3	2	1	1
4	2	4	1
5	3	1	1

## Bảng employees

<b>id</b>	<b>name</b>	<b>department</b>
1	David	Sales
2	Emma	Support
3	Frank	Sales

## Bảng order\_assignments

<b>id</b>	<b>order_id</b>	<b>employee_id</b>
1	1	1
2	2	3
3	3	2

## Bài tập

1. **Tạo index cho bảng ‘customers’** trên cột ‘city’ để tăng tốc các truy vấn tìm kiếm khách hàng theo thành phố.
2. **Tạo index cho bảng ‘orders’** trên cột ‘customer\_id’ để tối ưu truy vấn JOIN giữa ‘orders’ và ‘customers’.
3. **Tạo UNIQUE index cho bảng ‘products’** trên cột ‘name’ để đảm bảo tên sản phẩm không bị trùng lặp.
4. **Tạo index trên bảng ‘order\_items’** cho cột ‘product\_id’ và ‘order\_id’ (multi-column index) để tối ưu truy vấn chi tiết đơn hàng.
5. **Kiểm tra hiệu suất truy vấn có và không có index** bằng cách tạo truy vấn SELECT trên bảng lớn (gợi ý: dùng ‘EXPLAIN’).
6. **Xoá index vừa tạo ở câu 1.**
7. **Tạo index cho cột ‘total’** của bảng ‘orders’ để tăng tốc sắp xếp theo tổng đơn hàng.
8. **Tạo index cho bảng ‘employees’** theo ‘department’, sau đó truy vấn để lấy tất cả nhân viên thuộc ‘Sales’.
9. **So sánh tốc độ SELECT** có và không có index trên bảng ‘products’ khi tìm theo ‘price > 1000’.
10. **Tạo index trên bảng ‘orders’** theo cột ‘order\_date’ để tối ưu các truy vấn lấy đơn hàng gần đây.

# Trực Quan Hóa Dữ Liệu Không Gian với Folium

*Hoàng-Nguyễn Vũ*

## 1 Lý thuyết

Trong lĩnh vực khoa học dữ liệu và phân tích không gian địa lý (geospatial analysis), việc trực quan hóa dữ liệu trên bản đồ đóng vai trò quan trọng trong việc hỗ trợ con người hiểu rõ các mô hình, phân bố và xu hướng không gian. Đặc biệt trong bối cảnh dữ liệu ngày càng đa dạng và gắn liền với vị trí địa lý (ví dụ: dữ liệu GPS, bản đồ nhiệt, vùng dịch bệnh), việc sử dụng các công cụ trực quan như bản đồ tương tác giúp nâng cao hiệu quả trình bày và ra quyết định dựa trên dữ liệu.

## 2 Folium: Công cụ trực quan hóa dữ liệu không gian

### Khái quát

Folium là một thư viện Python mã nguồn mở, xây dựng trên nền tảng JavaScript Leaflet.js, cho phép người dùng tạo các bản đồ tương tác trực tiếp từ môi trường Python. Với cú pháp đơn giản nhưng linh hoạt, Folium hỗ trợ thể hiện đa dạng các loại dữ liệu không gian như: điểm (marker), vùng (polygon), đường đi, dữ liệu dưới định dạng GeoJSON, và bản đồ choropleth (bản đồ tô màu theo giá trị).

## 3 Kiến trúc hoạt động của Folium

Thư viện Folium hoạt động theo ba nguyên lý chính:

1. Khởi tạo một đối tượng bản đồ tại vị trí địa lý cụ thể thông qua `folium.Map()`.
2. Thêm các lớp (layer) dữ liệu không gian: điểm, vùng, dữ liệu định dạng GeoJSON, v.v.
3. Kết xuất bản đồ ra trình duyệt dưới dạng tệp HTML hoặc hiển thị trực tiếp trong môi trường Jupyter Notebook.

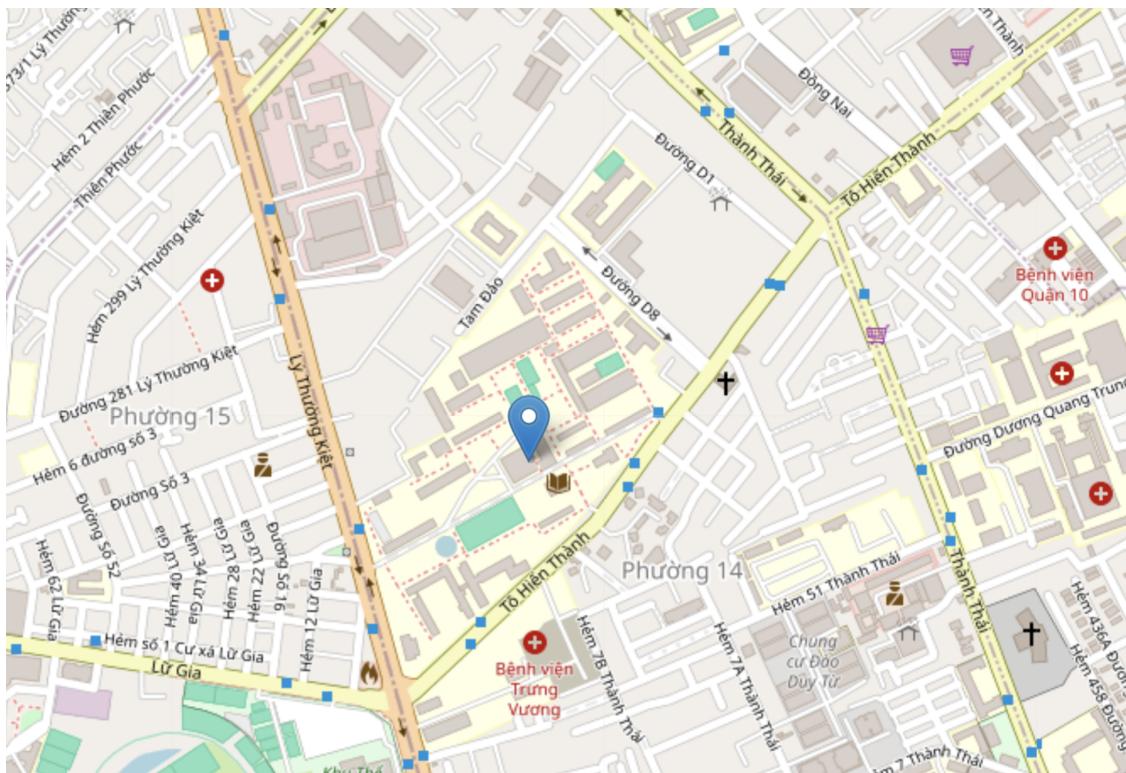
## 4 Các thành phần chính trong Folium

- `folium.Map()`: Tạo đối tượng bản đồ tại một vị trí cụ thể với các tùy chọn như mức zoom, loại tile nền.
- `folium.Marker()`: Đánh dấu một vị trí bằng biểu tượng, có thể kèm chú thích (popup).
- `folium.Circle()` và `folium.CircleMarker()`: Thể hiện vùng ảnh hưởng quanh điểm cụ thể.

- `folium.Polygon()`: Vẽ đa giác biểu thị một khu vực trên bản đồ.
  - `folium.GeoJson()`: Hiển thị dữ liệu không gian từ file GeoJSON.
  - `folium.Choropleth()`: Tạo bản đồ tô màu thể hiện giá trị định lượng theo khu vực.

## 5 Ví dụ: Vẽ bản đồ vị trí đơn giản

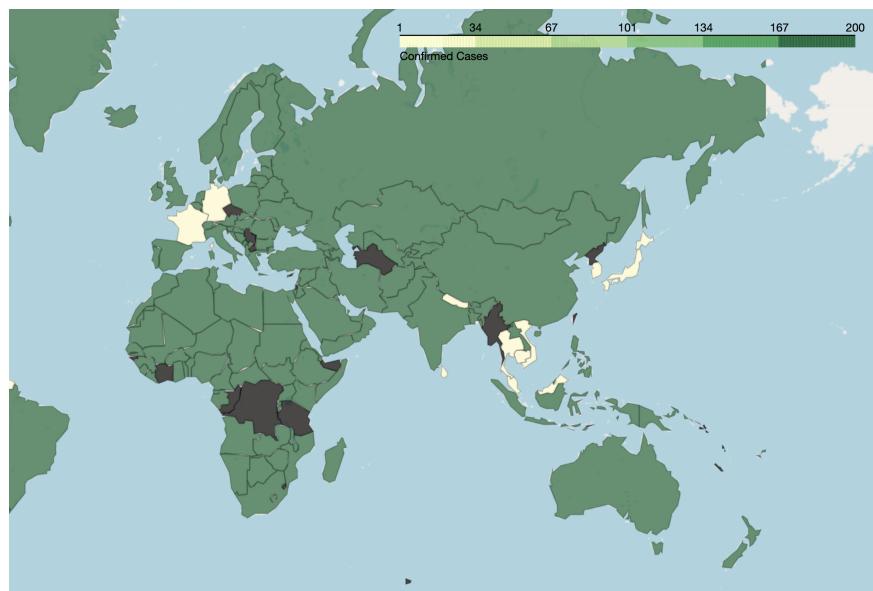
```
1 import folium
2
3 # Khởi tạo bản đồ tại tọa độ trung tâm TP. Hồ Chí Minh
4 hcmut_map = folium.Map(location=[10.762622, 106.660172], zoom_start=13)
5
6 # Đánh dấu vị trí cụ thể với popup
7 folium.Marker(
8     [10.762622, 106.660172],
9     popup="Trường ĐH Bách Khoa"
10 ).add_to(hcmut_map)
11
12 # Xuất bản đồ
13 hcmut_map
```



## 6 Trực quan hóa bản đồ Choropleth

Bản đồ choropleth là hình thức trực quan hóa dữ liệu định lượng trên bản đồ bằng cách tô màu các vùng (thường theo địa giới hành chính) dựa trên giá trị dữ liệu.

```
1 import pandas as pd
2 import folium
3 from folium.plugins import HeatMap
4
5 covid_data = pd.read_csv("covid_19_clean_complete.csv")
6
7 # Tạo bản đồ thế giới
8 world_map = folium.Map(location=[20, 0], zoom_start=4)
9
10 # Thêm Choropleth layer cho số ca nhiễm
11 folium.Choropleth(
12     geo_data='https://raw.githubusercontent.com/python-visualization/
13     folium/master/examples/data/world-countries.json',
14     name='choropleth',
15     data=covid_data,
16     columns=['Country/Region', 'Confirmed'],
17     key_on='feature.properties.name',
18     fill_color='YlGn',
19     fill_opacity=0.7,
20     line_opacity=0.2,
21     legend_name='Confirmed Cases'
22 ).add_to(world_map)
23
24 # Xuất bản đồ thế giới với Choropleth
25 world_map
```



## 7 So sánh với các thư viện khác

- **Matplotlib và Seaborn:** Không hỗ trợ tương tác và trực quan hóa bản đồ.
- **Plotly:** Có hỗ trợ bản đồ tương tác nhưng phức tạp hơn khi làm việc với dữ liệu GeoJSON hoặc dữ liệu hành chính Việt Nam.
- **Folium:** Ưu việt trong xử lý bản đồ nền, marker, tích hợp GeoJSON dễ dàng, phù hợp cho trình bày báo cáo trực tuyến.

## 8 Ứng dụng thực tiễn

- Phân tích dữ liệu dịch tễ học (COVID-19, sốt xuất huyết).
- Trực quan hóa thông tin kinh doanh theo khu vực.
- Bản đồ rủi ro thiên tai, môi trường.
- Hệ thống gợi ý địa điểm trong các ứng dụng bản đồ.

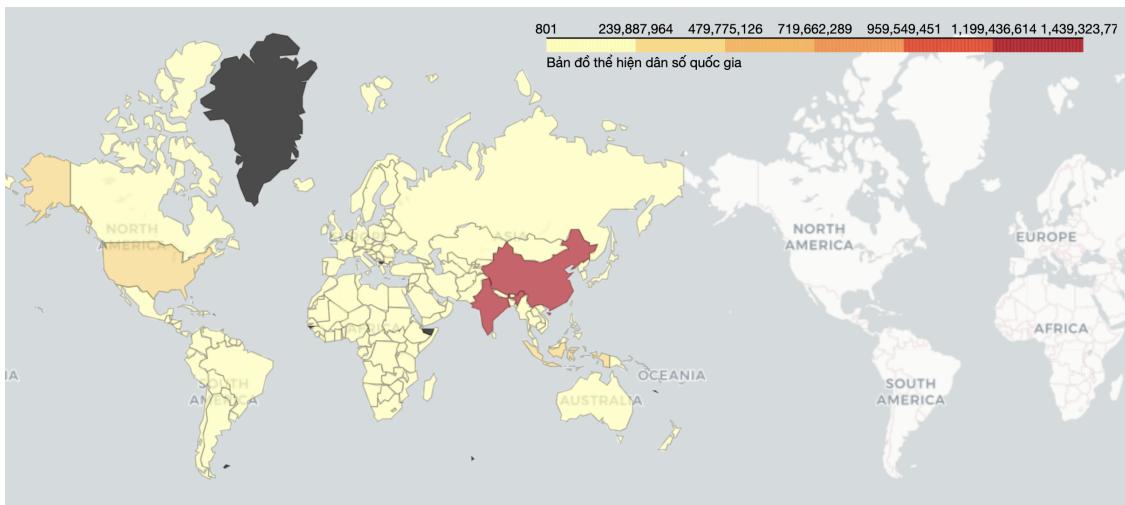
## 9 Kết luận

Thư viện Folium là một công cụ hiệu quả và dễ sử dụng trong việc trực quan hóa dữ liệu không gian với Python. Với khả năng tích hợp bản đồ tương tác, hỗ trợ nhiều định dạng dữ liệu không gian và khả năng mở rộng linh hoạt, Folium phù hợp với các bài toán phân tích địa lý trong thực tế cũng như giảng dạy.

## 10 Bài tập

Cho dữ liệu `world_countries_population_2020.csv` và `world-countries.json`, thực hiện các yêu cầu sau:

1. **Đọc dữ liệu** `world_countries_population_2020.csv`, hiển thị thông tin chung của dữ liệu bao gồm: `head`, `tail`, `info`, `describe`.
2. **Chuyển đổi kiểu dữ liệu** của các cột `Population` và `Lan Area (km2)` sang kiểu số.
3. **Tạo bản đồ** có kiểu nền `cartodbpositron`, với tâm bản đồ là Ấn Độ (`location = [20.5937, 78.9629]`) và mức zoom `zoom_start = 3`, gợi ý kết quả như hình sau:



# Chuẩn hóa cơ sở dữ liệu – First Normal Form (1NF)

*Hoàng-Nguyễn Vũ*

## 1 Lý thuyết: Chuẩn hóa 1NF

### 1.1 Định nghĩa học thuật

Một bảng (quan hệ) được coi là ở **chuẩn thứ nhất** (First Normal Form – 1NF) nếu:

- Mỗi ô trong bảng chứa một **giá trị nguyên tử** (atomic value).
- Không chứa danh sách, tập hợp, mảng trong một cột.
- Các cột có kiểu dữ liệu nhất quán và có tên duy nhất.

### 1.2 Ví dụ vi phạm 1NF

student_id	name	phone_numbers
1	An Nguyen	0901234567, 0987654321
2	Binh Tran	0901112222
3	Linh Pham	0933334444, 0944445555

**Lỗi:** Cột phone\_numbers chứa nhiều giá trị, vi phạm tính nguyên tử.

### 1.3 Cách chuẩn hóa về 1NF

Tách bảng:

- Students(student\_id, name)
- PhoneNumbers(student\_id, phone\_number)

Dữ liệu sau khi chuẩn hóa:

Students	
student_id	name
1	An Nguyen
2	Binh Tran
3	Linh Pham

PhoneNumbers	
student_id	phone_number
1	0901234567
1	0987654321
2	0901112222
3	0933334444
3	0944445555

## 2 Bài tập

Một công ty chứng khoán đang lưu trữ thông tin giao dịch của khách hàng trong bảng sau:

Bảng: Transactions\_Raw

transaction_id	customer_name	stock_codes	transaction_date
1	Nguyen Van A	VNM, FPT, MWG	2024-12-01
2	Le Thi B	SSI	2024-12-02
3	Tran Van C	VCB, TCB	2024-12-03

**Câu 1.** Phân tích và chỉ ra điểm nào trong bảng Transactions\_Raw vi phạm chuẩn 1NF? Giải thích lý do vi phạm.

**Câu 2.** Hãy thiết kế lại bảng dữ liệu để đảm bảo tuân thủ chuẩn 1NF.

**Câu 3.** Viết mô tả các bảng mới (tên bảng, tên các trường, khóa chính, khóa ngoại nếu có).

**Câu 4.** Sau khi chuẩn hóa, bạn hãy:

- Thêm dữ liệu vào các bảng mới theo đúng chuẩn 1NF dựa vào bảng Transactions\_Raw.
- Truy vấn danh sách khách hàng và các mã cổ phiếu tương ứng mà họ đã giao dịch.
- Cập nhật tên khách hàng Nguyen Van A thành Nguyen V. A.
- Xóa mã cổ phiếu MWG khỏi giao dịch của khách hàng Nguyen V. A.

**Câu 5.** Câu hỏi nâng cao:

- Viết truy vấn đếm số lượng mã cổ phiếu mỗi khách hàng đã giao dịch.
- Tìm khách hàng giao dịch nhiều mã cổ phiếu nhất.

# Chuẩn hóa cơ sở dữ liệu – Chuẩn thứ hai (2NF)

*Hoàng-Nguyễn Vũ*

## 1 Lý thuyết: Chuẩn thứ hai (Second Normal Form – 2NF)

### 1.1 Mục tiêu chuẩn hóa

Chuẩn hóa cơ sở dữ liệu giúp:

- Tránh dư thừa dữ liệu (data redundancy)
- Loại bỏ bất thường khi cập nhật (update anomalies)
- Bảo trì và mở rộng hệ thống dễ dàng

### 1.2 Định nghĩa 2NF

Một quan hệ (bảng) được coi là đạt **chuẩn thứ hai – Second Normal Form (2NF)** nếu:

- Đã đạt chuẩn 1NF
- Không có thuộc tính không khoá nào phụ thuộc vào một phần của khoá chính

### 1.3 Phụ thuộc từng phần là gì?

Khi khoá chính là một **tổ hợp nhiều cột**, nếu một thuộc tính chỉ phụ thuộc vào một phần của khoá chính → gọi là **phụ thuộc từng phần** → vi phạm 2NF.

### 1.4 Hệ quả khi vi phạm 2NF

- Gây dư thừa dữ liệu (lưu nhiều lần tên giống nhau)
- Khó cập nhật khi thông tin thay đổi (ví dụ: phải sửa tên khách hàng ở nhiều dòng)

### 1.5 Ví dụ vi phạm 2NF

Bảng: Enrollments\_Raw

student_id	course_id	student_name	course_name
1	C01	An Nguyen	SQL Basics
2	C02	Binh Tran	Python Intro
1	C02	An Nguyen	Python Intro

**Phân tích:**

- Khóa chính là: (`student_id`, `course_id`)
- `student_name` chỉ phụ thuộc vào `student_id`
- `course_name` chỉ phụ thuộc vào `course_id`
- → Vi phạm 2NF

## 1.6 Cách chuẩn hoá về 2NF

- Tách bảng thành các bảng nhỏ sao cho mỗi bảng chỉ chứa thuộc tính phụ thuộc hoàn toàn vào khóa chính.
- Tránh lặp lại thông tin không cần thiết.

**Bảng sau chuẩn hoá:**

- `Students(student_id, student_name)`
- `Courses(course_id, course_name)`
- `Enrollments(student_id, course_id)`

## 2 Bài tập

Một nhóm phát triển hệ thống AI Agent sử dụng LLM lưu trữ dữ liệu như sau:

**Bảng:** `AgentTasks_Raw`

<code>agent_id</code>	<code>agent_name</code>	<code>llm_model</code>	<code>task_id</code>	<code>task_name</code>	<code>task_type</code>
A01	Claude Agent	Claude 3	T01	Summarization	NLP
A02	GPT Assistant	GPT-4	T02	SQL Generation	Code
A01	Claude Agent	Claude 3	T03	Sentiment Analysis	NLP
A03	Gemini Helper	Gemini 1.5	T01	Summarization	NLP

**Câu 1.** Xác định khóa chính của bảng `AgentTasks_Raw`.

**Câu 2.** Bảng này có vi phạm 2NF không? Nếu có, chỉ rõ thuộc tính nào phụ thuộc từng phần.

**Câu 3.** Hãy chuẩn hoá bảng trên về chuẩn 2NF.

**Câu 4.** Mô tả các bảng sau chuẩn hoá: tên bảng, các cột, khóa chính và khóa ngoại.

**Câu 5.** Câu hỏi truy vấn:

- Lấy danh sách các tác vụ và tên Agent thực hiện.
- Lấy danh sách các mô hình LLM đang được sử dụng.
- Dếm số tác vụ thuộc mỗi loại (`task_type`).
- Tìm các Agent sử dụng LLM có tên bắt đầu bằng chữ G.

# Chuẩn hóa cơ sở dữ liệu – Chuẩn thứ ba (3NF)

*Hoàng-Nguyễn Vũ*

## 1 Lý thuyết: Chuẩn thứ ba (Third Normal Form – 3NF)

Chuẩn hóa cơ sở dữ liệu (database normalization) là quá trình tổ chức cấu trúc bảng trong cơ sở dữ liệu quan hệ nhằm:

- Giảm thiểu sự dư thừa dữ liệu (data redundancy)
- Loại bỏ các bất thường (anomalies) khi thêm, sửa, xoá dữ liệu
- Cải thiện khả năng mở rộng và bảo trì hệ thống

Quá trình chuẩn hóa thường trải qua các mức: 1NF → 2NF → 3NF → BCNF → 4NF...

## 2 Ôn tập các chuẩn trước 3NF

### 2.1 1NF – First Normal Form

- Mỗi bảng có khóa chính xác định duy nhất từng dòng.
- Mỗi ô trong bảng chỉ chứa một giá trị nguyên tử (atomic).
- Không chứa danh sách, tập hợp, hoặc mảng trong một ô.

### 2.2 2NF – Second Normal Form

- Đã đạt chuẩn 1NF.
- Không có thuộc tính không khóa nào phụ thuộc một phần vào khóa chính (áp dụng khi khóa là tổ hợp nhiều thuộc tính).

## 3 Chuẩn thứ ba – Third Normal Form (3NF)

### 3.1 Định nghĩa chính thức

Một bảng được coi là đạt **Third Normal Form (3NF)** nếu:

1. Bảng đã đạt chuẩn 2NF.

2. Mọi thuộc tính không khóa đều **phụ thuộc trực tiếp vào khóa chính**, tức là:  
Không tồn tại phụ thuộc bắc cầu (transitive dependency) dưới dạng:

Nếu  $A \rightarrow B$  và  $B \rightarrow C \Rightarrow A \rightarrow C$

Trong đó:

- A là khóa chính (hoặc khóa ứng viên),
- B không phải là khóa chính,
- C là thuộc tính không khóa.

### 3.2 Phụ thuộc bắc cầu là gì?

Là mối quan hệ gián tiếp giữa khóa chính và thuộc tính không khóa thông qua một thuộc tính trung gian.

Ví dụ:

- Khóa chính: emp\_id
- $\text{emp\_id} \rightarrow \text{dept\_id}$  và  $\text{dept\_id} \rightarrow \text{dept\_name}$
- $\rightarrow \text{emp\_id} \rightarrow \text{dept\_name}$  là phụ thuộc bắc cầu

### 3.3 Hết quả của vi phạm 3NF

- Dữ liệu bị dư thừa (ví dụ: tên phòng ban lặp lại nhiều lần)
- Dễ xảy ra lỗi khi cập nhật (cập nhật sai tên phòng ban)
- Khó mở rộng và duy trì

## 4 Ví dụ minh họa

Bảng: Employees \_ Raw

emp_id	emp_name	dept_id	dept_name
E01	An Nguyen	D01	Sales
E02	Binh Tran	D02	HR
E03	Lan Pham	D01	Sales

Phân tích:

- Khóa chính là emp\_id
- dept\_name phụ thuộc vào dept\_id, mà dept\_id phụ thuộc vào emp\_id
- dept\_name phụ thuộc bắc cầu vào emp\_id

**Cách chuẩn hóa:**

- Tách thành 2 bảng:
  1. Employees(emp\_id, emp\_name, dept\_id)
  2. Departments(dept\_id, dept\_name)

**5 Ghi nhớ 3 bước chuẩn hóa cơ bản**

- **1NF:** Mỗi ô chứa giá trị nguyên tử
- **2NF:** Loại bỏ phụ thuộc từng phần vào khoá tổng hợp
- **3NF:** Loại bỏ phụ thuộc bắc cầu vào khoá

**VI. Khi nào cần dùng 3NF?**

- Khi hệ thống có nhiều bảng chứa dữ liệu mô tả như tên khách hàng, tên phòng ban, tên sản phẩm...
- Khi một cột không khoá mô tả thuộc tính của một cột không khoá khác → nên chuẩn hóa.
- Khi muốn giảm dư thừa và lỗi cập nhật.

**6 Bài tập**

Một công ty logistics lưu trữ thông tin hàng hóa, kho lưu trữ, vị trí, khách hàng và nhân viên phụ trách trong cùng một bảng như sau:

**Record R001**

record_id	R001
item_code	IT001
item_name	Motor Oil 5L
quantity	50
warehouse_id	WH01
warehouse_name	Main Warehouse
location	Shelf A1
customer_id	C001
customer_name	Mekong Corp
staff_id	S01
staff_name	Hoa Nguyen

**Record R002**

record_id	R002
item_code	IT002
item_name	Engine Cleaner
quantity	30
warehouse_id	WH01
warehouse_name	Main Warehouse
location	Shelf B2
customer_id	C002
customer_name	Vina Auto
staff_id	S02
staff_name	Khang Tran

## Record R003

record_id	R003
item_code	IT001
item_name	Motor Oil 5L
quantity	40
warehouse_id	WH02
warehouse_name	North Depot
location	Shelf C3
customer_id	C001
customer_name	Mekong Corp
staff_id	S01
staff_name	Hoa Nguyen

## Record R004

record_id	R004
item_code	IT003
item_name	Battery 12V
quantity	25
warehouse_id	WH01
warehouse_name	Main Warehouse
location	Shelf A1
customer_id	C003
customer_name	Delta Co
staff_id	S02
staff_name	Khang Tran

Câu 1. Xác định khóa chính của bảng StorageRecords\_Raw.

Câu 2. Phân tích các vi phạm chuẩn hóa:

- a. Bảng trên đã đạt chuẩn 1NF chưa? Nếu chưa, cần điều chỉnh gì?
- b. Có thuộc tính nào vi phạm 2NF không? Nếu có, là những thuộc tính nào?
- c. Có thuộc tính nào vi phạm 3NF không? Nếu có, là những thuộc tính nào và vì sao?

Câu 3. Thực hiện chuẩn hóa đến 3NF:

- a. Trình bày các bảng sau khi chuẩn hóa.
- b. Ghi rõ khóa chính và khóa ngoại (nếu có) cho từng bảng.
- c. Tối thiểu có 4 bảng, khuyến khích thiết kế 5 bảng.

Câu 4. Thiết kế mô hình dữ liệu (mô tả dạng bảng hoặc sơ đồ ERD nếu có thể).

Sau khi chuẩn hóa, hãy viết truy vấn SQL (hoặc mô tả logic truy vấn) cho các yêu cầu sau:

- 4.1. Liệt kê tổng số lượng từng loại hàng hóa đang lưu trong từng kho.
- 4.2. Tìm tên khách hàng và các mặt hàng họ đã gửi kho.
- 4.3. Liệt kê danh sách nhân viên đang phụ trách các lô hàng lưu trữ tại kho có tên là “Main Warehouse”.
- 4.4. Tìm vị trí (location) lưu trữ của từng mặt hàng trong từng kho.
- 4.5. Dếm số khách hàng khác nhau đang gửi hàng tại mỗi kho.

# Tổng Quan Các Loại JOIN Trong SQL

*Hoàng-Nguyễn Vũ*

## 1 1. JOIN là gì?

Câu lệnh JOIN trong SQL dùng để kết hợp dữ liệu từ nhiều bảng dựa trên điều kiện chung (thường là khóa ngoại).

## 2 2. Các loại JOIN phổ biến

Loại JOIN	Mô tả	Dữ liệu giữ lại
INNER JOIN	Chỉ lấy các bản ghi khớp ở cả hai bảng	Cả hai bảng
LEFT JOIN	Lấy tất cả bản ghi ở bảng bên trái, kể cả khi không khớp	Bảng trái
RIGHT JOIN	Lấy tất cả bản ghi ở bảng bên phải, kể cả khi không khớp	Bảng phải
FULL OUTER JOIN	Lấy tất cả bản ghi từ cả hai bảng, nếu không có thì NULL	Cả hai bảng
CROSS JOIN	Nhân chéo giữa hai bảng (mọi tổ hợp có thể)	Không cần điều kiện

## 3 3. Ví dụ minh họa

### 3.1 Giả sử có 2 bảng

Bảng Customers:

CustomerID	Name
1	Alice
2	Bob
3	Charlie

Bảng Accounts:

AccountID	CustomerID	Balance
A1	1	500
A2	2	1500
A3	4	2000

### 3.2 INNER JOIN

```

1 SELECT c.Name, a.AccountID, a.Balance
2 FROM Customers c
3 INNER JOIN Accounts a ON c.CustomerID = a.CustomerID;
  
```

**Kết quả:**

Name	AccountID	Balance
Alice	A1	500
Bob	A2	1500

### 3.3 LEFT JOIN

```

1 SELECT c.Name, a.AccountID, a.Balance
2 FROM Customers c
3 LEFT JOIN Accounts a ON c.CustomerID = a.CustomerID;

```

**Kết quả:**

Name	AccountID	Balance
Alice	A1	500
Bob	A2	1500
Charlie	NULL	NULL

### 3.4 RIGHT JOIN

```

1 SELECT c.Name, a.AccountID, a.Balance
2 FROM Customers c
3 RIGHT JOIN Accounts a ON c.CustomerID = a.CustomerID;

```

**Kết quả:**

Name	AccountID	Balance
Alice	A1	500
Bob	A2	1500
NULL	A3	2000

### 3.5 FULL OUTER JOIN

```

1 SELECT c.Name, a.AccountID, a.Balance
2 FROM Customers c
3 FULL OUTER JOIN Accounts a ON c.CustomerID = a.CustomerID;

```

**Kết quả:**

Name	AccountID	Balance
Alice	A1	500
Bob	A2	1500
Charlie	NULL	NULL
NULL	A3	2000

### 3.6 CROSS JOIN

```

1 SELECT c.Name, a.AccountID
2 FROM Customers c
3 CROSS JOIN Accounts a;

```

**Kết quả:** Mọi tổ hợp giữa khách hàng và tài khoản.

## 4 4. Bài tập thực hành - Chủ đề Banking

Dữ liệu mẫu:

Bảng Customers:

CustomerID	Name	City
1	Alice	Hanoi
2	Bob	HCM
3	Charlie	Danang
4	Diana	Hanoi

Bảng Accounts:

AccountID	CustomerID	Balance
A1	1	500
A2	1	1000
A3	2	1500
A4	5	300

### Câu hỏi luyện tập

- Viết truy vấn liệt kê tất cả khách hàng và các tài khoản nếu có.
- Viết truy vấn chỉ hiển thị khách hàng có ít nhất một tài khoản.
- Viết truy vấn hiển thị tất cả tài khoản, kể cả không biết thuộc khách hàng nào.
- Viết truy vấn liệt kê toàn bộ kết hợp giữa khách hàng và tài khoản (mọi tổ hợp).
- Tìm khách hàng không có tài khoản.
- Tìm tài khoản không thuộc khách hàng nào.

# Bài Tập Thiết Kế Cơ Sở Dữ Liệu Hệ Thống Quản Lý Bán Hàng Quần Áo

Hoàng-Nguyễn Vũ

## I. Yêu cầu nghiệp vụ

Hệ thống quản lý bán hàng quần áo cần đáp ứng các yêu cầu nghiệp vụ sau:

- Quản lý danh mục sản phẩm: mỗi danh mục có mã và tên danh mục (ví dụ: Áo, Quần, Váy, Phụ kiện).
- Quản lý sản phẩm: mỗi sản phẩm có mã, tên, giá bán, thuộc một danh mục.
- Quản lý kho hàng: mỗi sản phẩm có thẻ có nhiều biến thể (màu sắc, kích thước), mỗi biến thể có số lượng tồn kho riêng.
- Quản lý khách hàng: mỗi khách hàng có họ tên, số điện thoại, địa chỉ và ngày tạo tài khoản.
- Quản lý đơn hàng: mỗi đơn hàng gắn với một khách hàng, có ngày tạo và trạng thái (Đã đặt, Đã giao, Đã huỷ).
- Quản lý chi tiết đơn hàng: mỗi đơn hàng có thẻ bao gồm nhiều sản phẩm (biến thể), kèm số lượng và giá tại thời điểm đặt hàng.
- Yêu cầu báo cáo tổng hợp: thống kê doanh thu theo ngày, theo sản phẩm, theo khách hàng.
- Yêu cầu thống kê bổ sung: thống kê số lượng khách hàng mới theo tháng, thống kê tỷ lệ đơn hàng theo trạng thái, thống kê tồn kho trung bình theo từng loại sản phẩm.

## II. Dữ liệu mẫu (Sample Data)

### 1. Danh mục sản phẩm

CategoryID	Name
1	Áo
2	Quần
3	Váy
4	Phụ kiện

## 2. Sản phẩm

ProductID	Name	CategoryID	Price
101	Áo sơ mi nam	1	300000
102	Quần jeans	2	500000
103	Váy xòe	3	450000

## 3. Biến thể sản phẩm

VariantID	ProductID	Color	Size	Stock
1	101	Trắng	M	20
2	101	Trắng	L	15
3	102	Xanh	M	10
4	103	Đỏ	S	5

## 4. Khách hàng

CustomerID	Name	Phone	Address	CreatedDate
1	Trần An	0909123456	Hà Nội	2024-01-10
2	Lê Bình	0909223456	TP.HCM	2024-02-05

## 5. Đơn hàng

OrderID	CustomerID	OrderDate	Status
1	1	2024-03-01	Đã giao
2	2	2024-03-02	Đã đặt

## 6. Chi tiết đơn hàng

OrderID	VariantID	Quantity	Price
1	1	2	300000
1	3	1	500000
2	4	1	450000

## III. Câu hỏi thực hành

- Viết các câu lệnh SQL để tạo các bảng phù hợp với mô hình dữ liệu trên (bao gồm khóa chính và khóa ngoại).
- Chèn toàn bộ dữ liệu mẫu ở mục II vào hệ cơ sở dữ liệu.

3. Liệt kê tất cả các đơn hàng, kèm theo tên khách hàng và tổng tiền của mỗi đơn hàng.
4. Liệt kê các sản phẩm có tồn kho dưới 10.
5. Tính tổng doanh thu theo từng ngày đặt hàng.
6. Liệt kê các khách hàng đã từng có đơn hàng.
7. Liệt kê sản phẩm bán chạy nhất (theo tổng số lượng bán).
8. Tính tổng doanh thu theo từng loại danh mục sản phẩm.
9. Hiển thị chi tiết của một đơn hàng cụ thể (gồm tên sản phẩm, màu, size, số lượng, đơn giá, tổng dòng).
10. Liệt kê các khách hàng chưa từng thực hiện đơn hàng nào.
11. Thống kê số lượng khách hàng mới theo từng tháng.
12. Thống kê số đơn hàng theo từng trạng thái (đã giao, đã đặt, đã huỷ).
13. Tính tồn kho trung bình theo từng loại sản phẩm.

# Statistics: Random Variable

*Hoàng-Nguyễn Vũ*

## 1 Biến Ngẫu Nhiên (Random Variable)

Một **biến ngẫu nhiên** là một hàm số ánh xạ mỗi kết quả của một thí nghiệm ngẫu nhiên thành một giá trị số thực:

$$X : \Omega \rightarrow \mathbb{R}$$

Trong AI, biến ngẫu nhiên được sử dụng để mô hình hóa các giá trị không chắc chắn như:

- Kết quả dự đoán (labels)
- Dữ liệu đầu vào hoặc nhiễu (noise)
- Biến tiềm ẩn trong mô hình sinh (latent variable)

## 2 Biến Ngẫu Nhiên Rời Rạc (Discrete Random Variable)

### Định nghĩa

Là biến chỉ nhận các giá trị rời rạc (đếm được). Xác suất của từng giá trị được biểu diễn qua hàm phân phối xác suất (PMF):

$$P(X = x_i) = p(x_i), \quad \sum_i p(x_i) = 1$$

### Ứng dụng trong AI

- Classification: nhãn lớp (0, 1, 2, ...)
- NLP: từ, token, nhãn sentiment
- Reinforcement Learning: action space

### Mã Python ví dụ

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 labels = np.random.choice(['cat', 'dog', 'rabbit'], size=1000, p=[0.4,
0.4, 0.2])
5 unique, counts = np.unique(labels, return_counts=True)

```

```

6
7 plt.bar(unique, counts / len(labels))
8 plt.title("Discrete Random Variable - Label Distribution")
9 plt.ylabel("Proportion")
10 plt.grid(True)
11 plt.show()

```

### 3 Biến Ngẫu Nhiên Liên Tục (Continuous Random Variable)

#### Định nghĩa

Là biến nhận giá trị từ một khoảng liên tục. Dùng hàm mật độ xác suất (PDF):

$$\int_{-\infty}^{\infty} f(x) dx = 1$$

$$P(a \leq X \leq b) = \int_a^b f(x) dx$$

#### Ứng dụng trong AI

- Regression (hồi quy): dự đoán giá, thời gian, nhiệt độ
- Noise injection trong GANs
- Latent variable trong VAE, Diffusion

#### Mã Python ví dụ

```

1 from scipy.stats import norm
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 x = np.linspace(-4, 4, 1000)
6 pdf = norm.pdf(x, loc=0, scale=1)
7
8 plt.plot(x, pdf, label="Normal PDF")
9 plt.fill_between(x, pdf, where=(x > -1) & (x < 1), alpha=0.4, label="Region [-1,1]")
10 plt.title("Continuous Random Variable - Normal Distribution")
11 plt.legend()
12 plt.grid(True)
13 plt.show()

```

## 4 Mô Hình Naive Bayes và Biến Ngẫu Nhiên

Naive Bayes giả định mỗi feature là một biến ngẫu nhiên độc lập:

$$P(y|X) \propto P(y) \prod_{i=1}^n P(x_i|y)$$

### Mã Python ví dụ

```

1 from sklearn.naive_bayes import GaussianNB
2 from sklearn.datasets import make_classification
3 from sklearn.model_selection import train_test_split
4 from sklearn.metrics import accuracy_score
5
6 X, y = make_classification(n_samples=1000, n_features=10, n_classes=3,
    random_state=42)
7 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
8
9 model = GaussianNB()
10 model.fit(X_train, y_train)
11 y_pred = model.predict(X_test)
12
13 print("Accuracy:", accuracy_score(y_test, y_pred))

```

## 5 Biến Ngẫu Nhiên Tiềm Ẩn trong VAE

Trong Variational Autoencoder, latent variable  $z \sim \mathcal{N}(\mu, \sigma^2)$ . Ta dùng:

$$z = \mu + \sigma \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1)$$

### Mã Python ví dụ

```

1 mu = np.zeros(10)
2 sigma = np.ones(10)
3 eps = np.random.normal(size=10)
4 z = mu + sigma * eps
5 print("Sampled latent vector z:", z)

```

## Bài tập:

### Bài 1: Sinh Nhãn Phân Loại Ngẫu Nhiên

Mô tả: Sinh 1000 nhãn thuộc 3 lớp `cat`, `dog`, `rabbit` với xác suất tương ứng là 0.4, 0.4, 0.2. In 10 nhãn đầu và tần suất xuất hiện.

### Bài 2: Sinh Noise Gaussian

Mô tả: Sinh 100 vector ngẫu nhiên (shape:  $(100, 10)$ ) từ phân phối chuẩn  $\mathcal{N}(0, 1)$ . Tính giá trị trung bình và độ lệch chuẩn của toàn mảng.

### Bài 3: Ước Lượng Xác Suất Từ Phân Phối Chuẩn

Mô tả: Mô phỏng 10.000 điểm từ phân phối chuẩn  $\mathcal{N}(0, 1)$  và ước lượng xác suất  $P(-1 < X < 1)$ .

### Bài 4: Mô Phỏng Phân Phối PMF

Mô tả: Sinh 1000 nhãn từ các lớp `positive`, `neutral`, `negative` với xác suất tương ứng [0.3, 0.5, 0.2]. Tính PMF thực tế từ dữ liệu sinh ra.

### Bài 5: Sampling Biến Tiềm Ẩn trong VAE

Mô tả: Áp dụng công thức *reparameterization trick* trong VAE:

$$z = \mu + \sigma \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1)$$

# Statistics: Expectation

*Hoàng-Nguyễn Vũ*

## 1. Lý thuyết về Expectation (Kỳ vọng)

Trong xác suất, **Expectation** của một biến ngẫu nhiên  $X$  là giá trị trung bình của  $X$  nếu thí nghiệm được lặp đi lặp lại nhiều lần.

### 1.1 Kỳ vọng của biến ngẫu nhiên rời rạc (Discrete)

$$\mathbb{E}[X] = \sum_i x_i \cdot P(X = x_i)$$

### 1.2 Kỳ vọng của biến liên tục (Continuous)

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x \cdot f(x) dx$$

Trong đó:

- $x_i$ : các giá trị có thể xảy ra của biến ngẫu nhiên
- $P(X = x_i)$ : xác suất của  $x_i$
- $f(x)$ : hàm mật độ xác suất

### 1.3 Ứng dụng trong AI

Expectation được sử dụng rộng rãi trong:

- Dự đoán giá trị trung bình đầu ra
- Học tăng cường: tính phần thưởng kỳ vọng
- Loss Function trong huấn luyện mô hình

### 1.4 Tính toán bằng NumPy

```

1 import numpy as np
2
3 x = np.array([1, 2, 3])
4 p = np.array([0.2, 0.5, 0.3])
5
6 expectation = np.sum(x * p)
7 print("E[X] =", expectation)

```

## 2. Bài tập thực hành Expectation với NumPy

### Bài 1: Tính kỳ vọng của biến rời rạc

Cho biến ngẫu nhiên  $X = [2, 4, 6, 8]$  với xác suất  $P = [0.1, 0.3, 0.4, 0.2]$ .

**Yêu cầu:** Viết code Python để tính  $\mathbb{E}[X]$ .

### Bài 2: Expectation từ phân phối chuẩn

Tạo dữ liệu gồm 10,000 mẫu tuân theo phân phối chuẩn  $\mu = 5, \sigma = 2$ . Tính giá trị kỳ vọng gần đúng từ dữ liệu.

```
1 data = np.random.normal(5, 2, 10000)
2 expectation = np.mean(data)
```

### Bài 3: Kỳ vọng của loss function trong AI

Cho:

```
1 y_pred = np.array([0.1, 0.4, 0.6, 0.9])
2 y_true = np.array([0, 0, 1, 1])
```

Tính kỳ vọng của Binary Cross-Entropy Loss:

$$\text{Loss} = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

### Bài 4: Kỳ vọng phần thưởng trong Reinforcement Learning

Giả sử phần thưởng từ 0 đến 10, phân phối:

```
1 rewards = np.random.choice(
2     np.arange(0, 11), size=100,
3     p=[0.05, 0.1, 0.1, 0.15, 0.1, 0.1, 0.1, 0.05, 0.1, 0.05]
4 )
```

Tính kỳ vọng của phần thưởng.

### Bài 5: So sánh kỳ vọng của hai mô hình AI

Dự đoán xác suất:

```
1 model1 = np.array([[0.2, 0.5, 0.3],
2                    [0.1, 0.7, 0.2]])
3
4 model2 = np.array([[0.3, 0.4, 0.3],
5                    [0.2, 0.6, 0.2]])
6
7 labels = np.array([1, 1]) # ground truth
```

Tính kỳ vọng cross-entropy loss cho từng mô hình và chọn mô hình tốt hơn.

# Statistics: Variance

*Hoàng-Nguyễn Vũ*

## 1. Lý thuyết về Variance (Phương sai)

### 1.1 Định nghĩa

Variance (phương sai) là đại lượng thể hiện mức độ phân tán của một biến ngẫu nhiên quanh giá trị kỳ vọng của nó.

### 1.2 Công thức

Biến ngẫu nhiên rời rạc:

$$\text{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2] = \sum_i (x_i - \mu)^2 \cdot P(X = x_i)$$

Phương sai mẫu:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

Phương sai tổng thể:

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$

Trong đó:

- $\mu = \mathbb{E}[X]$ : kỳ vọng của biến ngẫu nhiên
- $\bar{x}$ : trung bình mẫu
- $n$ : số lượng phần tử trong mẫu

### 1.3 Ứng dụng trong AI

- Variance giúp đánh giá sự ổn định của mô hình.
- Trong học máy, tồn tại khái niệm **bias-variance tradeoff** để cân bằng độ phức tạp mô hình.
- Dùng để chuẩn hóa dữ liệu (standardization) khi huấn luyện.

## 1.4 Tính toán bằng NumPy

```

1 import numpy as np
2
3 data = np.array([1, 2, 3, 4, 5])
4
5 population_var = np.var(data)
6 sample_var = np.var(data, ddof=1)
7
8 print("Population Variance:", population_var)
9 print("Sample Variance:", sample_var)

```

## 2. Bài tập thực hành Variance với NumPy

### Bài 1: Tính phương sai rời rạc

Cho biến ngẫu nhiên  $X = [1, 3, 5]$  với xác suất tương ứng  $P = [0.2, 0.5, 0.3]$ .

**Yêu cầu:** Tính kỳ vọng  $\mathbb{E}[X]$  và phương sai  $\text{Var}(X)$  bằng công thức rời rạc.

### Bài 2: Phân tích phương sai mẫu dữ liệu thực

Sinh ra dữ liệu gồm 1000 số thực từ phân phối chuẩn  $\mathcal{N}(0, 2^2)$ .

**Yêu cầu:** Tính phương sai mẫu và phương sai tổng thể. So sánh với giá trị lý thuyết.

### Bài 3: So sánh độ ổn định giữa hai mô hình AI

```

1 model_a_scores = np.array([0.8, 0.7, 0.9, 0.75, 0.85])
2 model_b_scores = np.array([0.6, 0.4, 0.9, 0.3, 0.8])

```

**Yêu cầu:** Tính phương sai và xác định mô hình nào ổn định hơn.

### Bài 4: Phân tích phân tán ảnh đầu vào

Giả sử bạn có trung bình pixel của 5 ảnh:

```

1 pixel_means = np.array([122, 120, 119, 123, 121])

```

**Yêu cầu:** Tính phương sai để đánh giá mức độ phân tán dữ liệu.

### Bài 5: Variance trong Reinforcement Learning

Một tác nhân nhận phần thưởng trong 10 lần thử nghiệm:

```

1 rewards = np.array([10, 9, 8, 10, 7, 6, 9, 10, 5, 8])

```

**Yêu cầu:** Tính phương sai phần thưởng để đánh giá độ ổn định của chiến lược.

# Statistics: Standard Deviation

*Hoàng-Nguyễn Vũ*

## 1. Lý thuyết về Standard Deviation

### 1.1 Định nghĩa

**Standard Deviation (Độ lệch chuẩn)** là căn bậc hai của phương sai, cho biết độ phân tán của dữ liệu quanh giá trị trung bình.

$$\text{Std}(X) = \sqrt{\text{Var}(X)} = \sqrt{\mathbb{E}[(X - \mathbb{E}[X])^2]}$$

Ký hiệu:

- $\sigma$ : độ lệch chuẩn tổng thể (population)
- $s$ : độ lệch chuẩn mẫu (sample)

### 1.2 Công thức

**Độ lệch chuẩn tổng thể:**

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2}$$

**Độ lệch chuẩn mẫu:**

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$$

### 1.3 Ứng dụng trong AI

- Dùng để chuẩn hóa dữ liệu đầu vào (standardization).
- Đánh giá độ ổn định của đầu ra mô hình.
- Tính tin cậy trong các mô hình xác suất.

### 1.4 Cách tính bằng NumPy

```

1 import numpy as np
2
3 data = np.array([1, 2, 3, 4, 5])
4
5 # Tổng thể
6 std_population = np.std(data)

```

```

7
8 # Mẫu
9 std_sample = np.std(data, ddof=1)
10
11 print("Population Std:", std_population)
12 print("Sample Std:", std_sample)

```

## 2. Bài tập thực hành Standard Deviation với NumPy

### Bài 1: Tính độ lệch chuẩn của biến rời rạc

Cho biến ngẫu nhiên  $X = [2, 4, 6]$  với xác suất  $P = [0.3, 0.4, 0.3]$ .

**Yêu cầu:** Tính kỳ vọng  $\mathbb{E}[X]$ , phương sai và độ lệch chuẩn của  $X$ .

### Bài 2: Độ lệch chuẩn từ phân phối chuẩn

Sinh 1000 số thực từ phân phối chuẩn  $\mathcal{N}(10, 3^2)$ .

**Yêu cầu:** Tính độ lệch chuẩn mẫu và tổng thể. So sánh với giá trị lý thuyết  $\sigma = 3$ .

### Bài 3: So sánh sự ổn định giữa hai mô hình AI

Cho dự đoán đầu ra:

```

1 model1 = np.array([0.85, 0.86, 0.84, 0.87])
2 model2 = np.array([0.9, 0.6, 0.95, 0.5])

```

**Yêu cầu:** Tính độ lệch chuẩn của mỗi mô hình và xác định mô hình nào ổn định hơn.

### Bài 4: Độ lệch chuẩn của ảnh trung bình (trong preprocessing)

Giả sử bạn có giá trị trung bình điểm ảnh của 10 ảnh huấn luyện:

```

1 pixel_values = np.array([123, 124, 122, 121, 125, 123, 124, 122, 120,
123])

```

**Yêu cầu:** Tính độ lệch chuẩn và đánh giá mức độ thay đổi.

### Bài 5: Độ lệch chuẩn phần thưởng trong Reinforcement Learning

Trong 20 lần thử nghiệm, agent nhận phần thưởng:

```

1 rewards = np.array([8, 7, 8, 9, 7, 8, 9, 8, 7, 8,
2 6, 8, 7, 8, 8, 7, 8, 9, 7, 8])

```

**Yêu cầu:** Tính độ lệch chuẩn để đánh giá độ ổn định của chiến lược học.

# Statistics: Correlation Coefficient

Hoàng-Nguyễn Vũ

## 1. Lý thuyết: Correlation Coefficient

### 1.1 Định nghĩa

Hệ số tương quan Pearson đo lường mức độ liên hệ tuyến tính giữa hai biến ngẫu nhiên  $X$  và  $Y$ .

$$r = \frac{\text{Cov}(X, Y)}{\sigma_X \cdot \sigma_Y}$$

Trong đó:

- $\text{Cov}(X, Y)$ : hiệp phương sai giữa  $X$  và  $Y$
- $\sigma_X, \sigma_Y$ : độ lệch chuẩn của  $X$  và  $Y$

### 1.2 Công thức tính từ dữ liệu

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

### 1.3 Ý nghĩa của hệ số $r$

- $r = 1$ : Tương quan tuyến tính dương hoàn hảo
- $r = -1$ : Tương quan tuyến tính âm hoàn hảo
- $r = 0$ : Không có tương quan tuyến tính

### 1.4 Ứng dụng trong AI

- Phân tích tương quan giữa đặc trưng và nhãn (feature selection)
- So sánh embedding vectors
- Tìm kiếm văn bản sử dụng TF-IDF và correlation

## 2. Bài tập Correlation Coefficient

**Bài 1. Tương quan tuyến tính hoàn hảo** Cho  $x = [1, 2, 3, 4, 5]$ ,  $y = [2, 4, 6, 8, 10]$ . Tính hệ số tương quan Pearson giữa  $x$  và  $y$ .

**Bài 2. Tương quan âm hoàn hảo** Cho  $x = [1, 2, 3, 4, 5]$ ,  $y = [10, 8, 6, 4, 2]$ . Tính hệ số tương quan và giải thích kết quả.

**Bài 3. Không tương quan tuyến tính** Tạo  $x \in [0, 10]$ ,  $y = \sin(x)$  với 100 điểm. Tính Pearson correlation giữa  $x$  và  $y$ .

**Bài 4. Tương quan giữa đặc trưng và nhãn** Cho feature = [1.1, 1.9, 3.2, 4.5, 5.1], label = [1.0, 2.0, 3.0, 4.1, 5.3]. Tính tương quan giữa feature và label.

**Bài 5. Tương quan trong bảng dữ liệu** Cho height = [150, 160, 170, 180, 190], weight = [50, 60, 70, 80, 90]. Tính tương quan giữa chiều cao và cân nặng.

**Bài 6. Embedding similarity** Cho  $\text{embed}_A = [0.3, 0.5, 0.7, 0.8]$ ,  $\text{embed}_B = [0.9, 1.4, 2.1, 2.4]$ . Tính tương quan giữa hai vectors.

**Bài 7. Tương quan ngẫu nhiên** Sinh hai vector ngẫu nhiên  $x, y \in [0, 1]$  gồm 100 giá trị. Tính Pearson correlation và nhận xét.

**Bài 8. Tương quan với nhiễu (noise)** Cho  $x = [0, 1, 2, \dots, 99]$ ,  $y_{\text{clean}} = x$ ,  $y_{\text{noisy}} = x + \text{noise}$ . So sánh tương quan giữa  $x$  và hai biến  $y$ .

**Bài 9. Dữ liệu thời gian: nhiệt độ và doanh số** Cho temperature = [22, 24, 23, 25, 26], sales = [100, 110, 105, 115, 120]. Tính tương quan giữa nhiệt độ và doanh số.

**Bài 10. Retrieval văn bản với TF-IDF + Pearson (không dùng thư viện)** Cho các văn bản:

```

1 doc1 = "deep learning for natural language processing"
2 doc2 = "transformer models improve language understanding"
3 doc3 = "convolutional neural networks for image classification"
4 query = "language models for text understanding"

```

Tính TF-IDF, sau đó tính hệ số tương quan Pearson giữa truy vấn và từng văn bản. Xếp hạng các văn bản theo mức độ liên quan.

# Streamlit

*Hoàng-Nguyễn Vũ*

## I. Lý thuyết về Streamlit

### 1. Streamlit là gì?

Streamlit là một thư viện Python mã nguồn mở, cho phép xây dựng giao diện Web tương tác nhanh chóng dành cho các dự án Machine Learning và Data Science.

### 2. Cài đặt

```
1 pip install streamlit
```

### 3. Cách chạy một ứng dụng Streamlit

```
1 streamlit run app.py
```

### 4. Ví dụ Streamlit cơ bản

```
1 import streamlit as st
2
3 st.title("Tiêu đề")
4 st.write("Đây là ứng dụng Streamlit đơn giản.")
```

### 5. Kết hợp Seaborn để vẽ biểu đồ

```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3 import streamlit as st
4
5 df = sns.load_dataset("tips")
6
7 fig, ax = plt.subplots()
8 sns.scatterplot(data=df, x="total_bill", y="tip", ax=ax)
9 st.pyplot(fig)
```

## II. Bài Tập Thực Hành

### Bài 1: Vẽ đồ thị hàm số cơ bản

**Yêu cầu:** Chọn 1 trong các hàm số  $\sin$ ,  $\cos$ ,  $\exp$ ,  $\log$  và vẽ biểu đồ trên đoạn  $[-10, 10]$ .

### Bài 2: So sánh 2 hàm số trên cùng một biểu đồ

**Yêu cầu:** Chọn hai hàm số bất kỳ trong số:  $\sin$ ,  $\cos$ ,  $\exp$ ,  $\log$ , và vẽ chúng trên cùng một biểu đồ.

### Bài 3: Vẽ đồ thị hàm bậc 2

**Yêu cầu:** Nhập hệ số  $a$ ,  $b$ ,  $c$  cho phương trình  $y = ax^2 + bx + c$  và vẽ đồ thị tương ứng.

### Bài 4: Tương tác với Slider để khảo sát đồ thị

**Yêu cầu:** Dùng `st.slider` để điều chỉnh giá trị của  $a$ ,  $b$ ,  $c$  và cập nhật đồ thị theo thời gian thực.

### Bài 5: Vẽ Heatmap cho hàm $z = x^2 + y^2$

**Yêu cầu:** Dùng `sns.heatmap` để vẽ biểu đồ nhiệt của hàm  $z = x^2 + y^2$ .

# Tạo Ứng Dụng Phân Tích Dữ Liệu với Streamlit và PygWalker

*Hoàng-Nguyễn Vũ*

## I. Lý thuyết

### 1. Streamlit là gì?

Streamlit là một thư viện mã nguồn mở của Python giúp bạn dễ dàng chuyển các đoạn mã Python thông thường thành ứng dụng web tương tác. Đây là công cụ lý tưởng cho các nhà khoa học dữ liệu và kỹ sư AI muốn:

- Trình bày kết quả phân tích một cách trực quan.
- Tạo giao diện nhập dữ liệu, chọn tham số mô hình.
- Triển khai ứng dụng mà không cần viết HTML/CSS/JS.

### 2. PygWalker là gì?

PygWalker (Python + Graphic Walker) là một công cụ giúp trực quan hóa dữ liệu tương tác trong Python. Nó cho phép bạn phân tích dữ liệu theo phong cách kéo-thả như Tableau.

#### Ưu điểm:

- Tương tác trực tiếp với DataFrame.
- Giao diện kéo-thả thân thiện.
- Không cần viết code để tạo biểu đồ.

### 3. Cài đặt thư viện

```
1 pip install streamlit pygwalker pandas
```

### 4. Các thành phần chính của ứng dụng

- `st.file_uploader()`: cho phép tải file CSV.
- `pandas.read_csv()`: đọc nội dung file.
- `st.dataframe()`: hiển thị dữ liệu dạng bảng.
- `pygwalker.walk()`: tạo giao diện EDA.
- `st.components.html()`: nhúng PygWalker vào Streamlit.

## 5. Lưu ý khi dùng PygWalker

- PygWalker tạo giao diện HTML nên cần nhúng thủ công.
- Nên kiểm tra dữ liệu sạch trước khi phân tích.
- PygWalker phù hợp với dataset có kích thước trung bình.

## II. Bài tập thực hành

### Bài 1: Tải file CSV và hiển thị dữ liệu

Yêu cầu:

- Cho phép người dùng tải file CSV.
- Đọc nội dung bằng pandas.
- Hiển thị dữ liệu đầu tiên và thông tin tổng quan.

```

1 import streamlit as st
2 import pandas as pd
3
4 st.title("Phân tích dữ liệu từ file CSV")
5
6 uploaded_file = st.file_uploader("Chọn file CSV", type=["csv"])
7 if uploaded_file is not None:
8     df = pd.read_csv(uploaded_file)
9     st.subheader("Dữ liệu ban đầu")
10    st.dataframe(df.head())
11    st.write("Số dòng:", df.shape[0])
12    st.write("Số cột:", df.shape[1])

```

### Bài 2: Thống kê mô tả dữ liệu

Yêu cầu:

- Hiển thị df.describe().
- Hiển thị kiểu dữ liệu (df.dtypes).
- Hiển thị số lượng giá trị thiếu trên mỗi cột.

```

1 if uploaded_file is not None:
2     st.subheader("Thông tin mô tả dữ liệu")
3     st.write("Các thống kê cơ bản:")
4     st.dataframe(df.describe())
5     st.write("Kiểu dữ liệu:")
6     st.write(df.dtypes)
7     st.write("Số giá trị thiếu:")
8     st.write(df.isnull().sum())

```

## Bài 3: Phân tích EDA bằng PygWalker

Yêu cầu:

- Tích hợp giao diện EDA PygWalker.
- Cho phép người dùng tương tác với dữ liệu (kéo-thả tạo biểu đồ).

```
1 import streamlit.components.v1 as components
2 import pygwalker as pyg
3
4 if uploaded_file is not None:
5     st.subheader("Phân tích dữ liệu tương tác với PygWalker")
6     pyg_html = pyg.walk(df, return_html=True)
7     components.html(pyg_html, height=1000, scrolling=True)
```

## Bài 4: Gợi ý mở rộng

- Cho phép người dùng chọn cột để lọc hoặc phân tích cụ thể.
- Vẽ biểu đồ tương quan giữa các cột số bằng Seaborn.
- Cho phép người dùng tải dữ liệu đã lọc về dưới dạng CSV.

# Tạo Ứng Dụng ChatGPT cơ bản với Streamlit

*Hoàng-Nguyễn Vũ*

## I. Lý thuyết

### 1. Streamlit là gì?

Streamlit là một thư viện mã nguồn mở của Python giúp bạn dễ dàng chuyển các đoạn mã Python thông thường thành ứng dụng web tương tác. Đây là công cụ lý tưởng cho các nhà khoa học dữ liệu và kỹ sư AI muốn:

- Trình bày kết quả phân tích một cách trực quan.
- Tạo giao diện nhập dữ liệu, chọn tham số mô hình.
- Triển khai ứng dụng mà không cần viết HTML/CSS/JS.

### 2. Mô hình hoạt động của ứng dụng ChatGPT với Streamlit

- Giao diện người dùng được xây dựng bằng Streamlit.
- Người dùng nhập tin nhắn vào hộp thoại.
- Tin nhắn được gửi tới API của OpenAI.
- Phản hồi từ API được hiển thị như một tin nhắn trả lời từ ChatGPT.
- Cuộc hội thoại được lưu trong session (bộ nhớ tạm thời).

### 3. Cài đặt thư viện cần thiết

```
1 pip install streamlit openai
```

### 4. Đăng ký API Key tại OpenAI

Tạo tài khoản tại <https://platform.openai.com/account/api-keys> và lấy API key để gọi GPT.

## II. Bài tập: Xây dựng khung chat đơn giản với GPT-4o

### 1. Khởi tạo ứng dụng Streamlit

```
1 import streamlit as st
2 import openai
3 import os
```

### 2. Nhập API Key (ẩn)

```
1 openai.api_key = st.secrets["OPENAI_API_KEY"]  
(Hoặc dùng: openai.api_key = os.getenv("OPENAI_API_KEY"))
```

### 3. Thiết kế giao diện và lưu session chat

```
1 st.title(" Chat với GPT-4o")
2
3 if "messages" not in st.session_state:
4     st.session_state.messages = []
5
6 # Hiển thị toàn bộ hội thoại
7 for msg in st.session_state.messages:
8     st.chat_message(msg["role"]).write(msg["content"])
```

### 4. Gửi tin nhắn và gọi API OpenAI

```
1 if prompt := st.chat_input("Bạn muốn hỏi gì?"):
2     st.chat_message("user").write(prompt)
3     st.session_state.messages.append({"role": "user", "content": prompt})
4
5     response = openai.ChatCompletion.create(
6         model="gpt-4o",
7         messages=st.session_state.messages
8     )
9
10    reply = response.choices[0].message.content
11    st.chat_message("assistant").write(reply)
12    st.session_state.messages.append({"role": "assistant", "content": reply})
```

# Minh họa Huấn luyện Linear Regression bằng Numpy và Streamlit

*Hoàng-Nguyễn Vũ*

## I. Lý thuyết cơ bản

### 1. Linear Regression là gì?

Hồi quy tuyến tính (Linear Regression) là một thuật toán học có giám sát (supervised learning) đơn giản nhưng mạnh mẽ dùng để dự đoán giá trị liên tục. Dạng cơ bản của hàm dự đoán là:

$$\hat{y} = \mathbf{X}\mathbf{w} + b$$

Trong đó:

- $\mathbf{X}$  là ma trận đặc trưng (mỗi hàng là một mẫu).
- $\mathbf{w}$  là vector trọng số.
- $b$  là hệ số chêch (bias).

Mục tiêu là tìm  $\mathbf{w}, b$  sao cho hàm măt mát (loss function) nhỏ nhất. Sử dụng:

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2$$

### 2. Gradient Descent với vector hóa

Sử dụng đạo hàm và cập nhật theo gradient descent:

$$\begin{aligned} \mathbf{w} &\leftarrow \mathbf{w} - \alpha \cdot \frac{2}{n} \mathbf{X}^T (\mathbf{X}\mathbf{w} + b - \mathbf{y}) \\ b &\leftarrow b - \alpha \cdot \frac{2}{n} \sum (\hat{y} - y) \end{aligned}$$

Trong đó  $\alpha$  là learning rate.

### 3. advertising.csv là gì?

Đây là dataset phổ biến trong thống kê marketing, chứa các cột:

- TV: ngân sách quảng cáo trên TV.
- Radio: ngân sách quảng cáo trên radio.
- Newspaper: ngân sách quảng cáo trên báo in.
- Sales: doanh số bán hàng (giá trị mục tiêu).

## II. Bài tập: Huấn luyện Linear Regression với Numpy + Streamlit

### 1. Đọc dữ liệu từ CSV

```

1 import pandas as pd
2
3 df = pd.read_csv("advertising.csv")
4 X = df[["TV", "Radio", "Newspaper"]].values
5 y = df["Sales"].values.reshape(-1, 1)

```

### 2. Chuẩn hoá dữ liệu đầu vào

```

1 X = (X - X.mean(axis=0)) / X.std(axis=0)

```

### 3. Hàm huấn luyện Linear Regression dùng Numpy vector hóa

```

1 import numpy as np
2
3 def train_linear_regression(X, y, lr=0.01, epochs=100):
4     # Your code here #

```

### 4. Tạo giao diện Streamlit để theo dõi training

```

1 import streamlit as st
2 import matplotlib.pyplot as plt
3
4 st.title("Huấn luyện Linear Regression với Numpy")
5
6 lr = st.slider("Learning rate", 0.001, 0.1, 0.01)
7 epochs = st.slider("Số epoch", 10, 1000, 200)
8
9 if st.button("Train model"):
10     w, b, loss_history = train_linear_regression(X, y, lr, epochs)
11
12     st.subheader("Loss theo epoch")
13     fig, ax = plt.subplots()
14     ax.plot(loss_history)
15     ax.set_xlabel("Epoch")
16     ax.set_ylabel("MSE Loss")
17     st.pyplot(fig)
18
19     st.write("Trọng số cuối cùng (w):", w.flatten())
20     st.write("Bias (b):", b)

```

# Lambda

*Hoàng-Nguyễn Vũ*

## 1 Lý Thuyết: Hàm lambda trong Python

### 1.1 1. Khái niệm

- `lambda` trong Python là cách tạo hàm nhỏ gọn, nhanh chóng, còn gọi là *anonymous function* (hàm ẩn danh).
- Không cần đặt tên cho hàm như khi dùng `def`.

Cú pháp:

```
1 lambda arguments: expression
```

- **arguments:** các biến đầu vào
- **expression:** biểu thức trả về (chỉ 1 dòng, không nhiều lệnh)

### 1.2 2. Ví dụ đơn giản

```
1 add = lambda x, y: x + y
2 print(add(3, 5)) # Output: 8
```

### 1.3 3. Khi nào nên dùng lambda?

- Khi cần viết hàm ngắn gọn, chỉ 1 dòng.
- Khi truyền hàm làm đối số cho các hàm khác như `map()`, `filter()`, `sorted()`, v.v.

Ví dụ:

```
1 numbers = [1, 2, 3, 4]
2 squared = list(map(lambda x: x**2, numbers))
3 print(squared) # Output: [1, 4, 9, 16]
```

### 1.4 4. So sánh lambda và def

Tiêu chí	<code>lambda</code>	<code>def</code>
Độ dài	Ngắn gọn	Dài hơn
Tên hàm	Không cần tên	Cần có tên
Nội dung	1 dòng expression	Nhiều dòng lệnh
Sử dụng	Hàm nhỏ, nhanh gọn	Hàm lớn, phức tạp

## 2 Bài Tập Thực Hành

### Bài 1

Tạo một hàm lambda tính bình phương một số.

**Input:** 4    **Output:** 16

### Bài 2

Tạo hàm lambda kiểm tra xem một số có lớn hơn 10 không.

**Input:** 15    **Output:** True

### Bài 3

Dùng map và lambda để nhân đôi từng phần tử trong list [1, 3, 5, 7].

**Output:** [2, 6, 10, 14]

### Bài 4

Dùng filter và lambda để lọc các số chẵn từ list [1, 2, 3, 4, 5, 6].

**Output:** [2, 4, 6]

### Bài 5

Sắp xếp danh sách các tuple theo phần tử thứ hai dùng lambda.

```
1 students = [('John', 80), ('Jane', 95), ('Dave', 90)]
```

**Kết quả mong muốn:**

```
1 [('John', 80), ('Dave', 90), ('Jane', 95)]
```

## 3 Bonus: Bài Tập Nâng Cao

### Bài 6

Viết hàm calculator nhận vào một phép toán (+, -, \*, /) và trả về một hàm lambda thực hiện phép toán đó.

**Ví dụ:**

```
1 add = calculator('+')
2 print(add(3, 4)) # Output: 7
```

**Gợi ý:**

```
1 def calculator(op):
2     if op == '+':
3         return lambda x, y: x + y
4     elif op == '-':
```

```
5     return lambda x, y: x - y
6 elif op == '*':
7     return lambda x, y: x * y
8 elif op == '/':
9     return lambda x, y: x / y
```

# Basic Computer Vision

*Hoàng-Nguyễn Vũ*

## 1 Lý Thuyết Cơ Bản

### 1.1 1. Đọc ảnh (cv2.imread)

Cú pháp:

```
1 import cv2
2 image = cv2.imread('path_to_image.jpg')
```

Ghi chú:

- Mặc định OpenCV đọc ảnh dưới dạng không gian màu BGR.
- Nếu ảnh không tìm thấy, cv2.imread sẽ trả về None.

### 1.2 2. Hiển thị ảnh (cv2.imshow) và Đợi phím nhấn

```
1 cv2.imshow('Window Name', image)
2 cv2.waitKey(0) # Đợi phím bất kỳ
3 cv2.destroyAllWindows() # Đóng tất cả cửa sổ
```

### 1.3 3. Lưu ảnh (cv2.imwrite)

```
1 cv2.imwrite('saved_image.jpg', image)
```

Ghi chú: Nếu đường dẫn không hợp lệ, ảnh sẽ không được lưu.

### 1.4 4. Chuyển đổi không gian màu (cv2.cvtColor)

Cú pháp:

```
1 converted_image = cv2.cvtColor(image, cv2.COLOR_CODE)
```

Các COLOR\_CODE phổ biến:

- cv2.COLOR\_BGR2RGB: BGR → RGB
- cv2.COLOR\_BGR2HSV: BGR → HSV
- cv2.COLOR\_BGR2GRAY: BGR → Grayscale
- cv2.COLOR\_RGB2BGR: RGB → BGR
- cv2.COLOR\_RGB2HSV: RGB → HSV

## 1.5 5. Các hệ màu cần biết

- **RGB (Red-Green-Blue):** Phổ biến trong xử lý ảnh tự nhiên, mỗi kênh từ 0 đến 255.
- **BGR (Blue-Green-Red):** OpenCV mặc định đọc ảnh theo thứ tự BGR.
- **HSV (Hue-Saturation-Value):** Phù hợp cho các bài toán tách màu.
- **HSB (Hue-Saturation-Brightness):** Tương tự HSV, chỉ khác tên gọi Value  $\approx$  Brightness.
- **Grayscale:** Ảnh xám, chỉ có độ sáng (1 kênh).

## 2 Bài Tập Thực Hành: Chọn ảnh bất kỳ

### 2.1 Bài 1: Đọc và hiển thị ảnh

Viết chương trình đọc file ảnh tên `input.jpg` và hiển thị ảnh ra màn hình.

**Gợi ý:** Sử dụng `cv2.imread()` và `cv2.imshow()`.

### 2.2 Bài 2: Lưu ảnh dưới tên khác

Viết chương trình lưu ảnh vừa đọc thành `output.jpg`.

**Gợi ý:** Sử dụng `cv2.imwrite()`.

### 2.3 Bài 3: Chuyển ảnh từ BGR sang RGB

Viết chương trình chuyển ảnh từ không gian BGR sang RGB.

**Gợi ý:** Sử dụng `cv2.cvtColor()` với `cv2.COLOR_BGR2RGB`.

### 2.4 Bài 4: Chuyển ảnh từ BGR sang HSV

Chuyển ảnh gốc từ BGR sang HSV và hiển thị kết quả.

**Gợi ý:** Sử dụng `cv2.COLOR_BGR2HSV`.

### 2.5 Bài 5: Chuyển ảnh sang Grayscale

Chuyển ảnh gốc sang ảnh đen trắng (grayscale) và lưu lại với tên `gray.jpg`.

**Gợi ý:** Sử dụng `cv2.COLOR_BGR2GRAY`.

## 2.6 Bài 6: So sánh ảnh RGB và HSV

Hiển thị ảnh gốc (RGB) và ảnh sau khi chuyển HSV cạnh nhau. Ghi nhận sự khác biệt.

Gợi ý:

- Chuyển đổi ảnh sang RGB và HSV.
- Dùng cv2.hconcat để ghép ảnh theo chiều ngang.

```
1 combined = cv2.hconcat([image_rgb, image_hsv])
2 cv2.imshow('Comparison', combined)
```

# Object Detection

Hoàng-Nguyễn Vũ

## 1 Lý Thuyết Cơ Bản

### 1.1 1. Giới thiệu về YOLOv11

- **YOLOv11** là một phiên bản mới của họ mô hình YOLO (*You Only Look Once*), nổi bật với tốc độ nhanh, độ chính xác cao, và khả năng generalize tốt hơn trên các bài toán phát hiện vật thể (object detection).
- YOLOv11 cải thiện trên các điểm sau:
  - Backbone mới hiệu quả hơn.
  - Neck và Head cải tiến để tối ưu hóa multi-scale feature fusion.
  - Kỹ thuật label assignment và augmentation tiên tiến.

### 1.2 2. Bài toán phát hiện Người và Mũ Bảo Hiểm

- Mục tiêu: Xác định vị trí (bounding box) và phân loại đối tượng là Người hoặc Mũ Bảo Hiểm trong ảnh hoặc video.
- Ứng dụng thực tế: An toàn giao thông, giám sát công trường, quản lý tuân thủ quy định.

### 1.3 3. Quy trình cơ bản sử dụng YOLOv11

1. **Chuẩn bị môi trường:** Cài đặt thư viện cần thiết như ultralytics, torch, opencv-python.
2. **Tải mô hình YOLOv11:** Dùng mô hình pretrained hoặc fine-tune thêm với dataset chuyên biệt.
3. **Dự đoán (Inference):** Nạp ảnh/video và dùng mô hình để phát hiện người và mũ bảo hiểm.
4. **Xử lý đầu ra:** Vẽ bounding boxes, gán nhãn đối tượng.

### 1.4 4. Một số chú ý khi detect Người và Mũ Bảo Hiểm

- Chọn ngưỡng confidence (`confidence threshold`) hợp lý, thường khoảng 0.25 - 0.5.
- Nếu cần phát hiện riêng biệt (ví dụ, người đội mũ vs. người không đội), cần fine-tune thêm mô hình hoặc thêm post-processing logic.

## 2 Bài Tập Thực Hành

Data tải tại đây

### Bài 1: Cài đặt thư viện

Hãy cài đặt các thư viện cần thiết cho bài tập:

```
1 pip install ultralytics opencv-python
```

Gợi ý: Sử dụng Google Colab hoặc local environment.

### Bài 2: Tải mô hình YOLOv11 pretrained

Viết đoạn mã tải mô hình YOLOv11 pretrained.

```
1 from ultralytics import YOLO  
2  
3 model = YOLO('yolov11.pt') # hoặc dùng phiên bản custom
```

### Bài 3: Fine-tune mô hình với dữ liệu mũ bảo hiểm riêng

- Thu thập hoặc sử dụng dataset như Helmet Detection Dataset.
- Chuẩn hóa dữ liệu theo định dạng YOLOv11 (.txt label files).
- Fine-tune mô hình với lệnh:

```
1 yolo train model=yolov11.pt data=helmet.yaml epochs=50 imgsz=640
```

### Bài 4: Phát hiện Người và Mũ Bảo Hiểm trên ảnh

Viết chương trình đọc một ảnh và phát hiện người và mũ bảo hiểm trong ảnh.

```
1 results = model.predict('input_image.jpg', conf=0.5)  
2  
3 # Hiển thị ảnh kết quả  
4 results[0].show()
```

**Yêu cầu:** Hiển thị rõ bounding box và nhãn tên đối tượng.

### Bài 5: Phát hiện Người và Mũ Bảo Hiểm trên video

Viết chương trình chạy detection trên file video.

```
1 results = model.predict(source='input_video.mp4', stream=True)  
2  
3 for result in results:  
4     frame = result.plot()  
5     # Hiển thị từng khung hình hoặc lưu lại
```

**Yêu cầu:** Hiển thị bounding boxes theo thời gian thực.

# Thuật toán Linear Search (Tìm kiếm tuyến tính)

*Hoàng-Nguyễn Vũ*

## 1. Khái niệm

**Linear Search** (tìm kiếm tuyến tính) là thuật toán tìm kiếm tuần tự phần tử trong mảng. Thuật toán sẽ duyệt từng phần tử từ trái sang phải, so sánh với giá trị cần tìm.

### Dặc điểm

- Không yêu cầu mảng phải sắp xếp.
- Độ phức tạp thời gian:
  - Trường hợp tốt nhất:  $O(1)$
  - Trường hợp xấu nhất:  $O(n)$

## 2. Giả mã (Pseudocode)

```
1 Input: array A of n elements, value x to find
2 Output: index of x if found, else -1
3
4 for i from 0 to n-1:
5     if A[i] == x:
6         return i
7 return -1
```

## 3. Cài đặt Python

```
1 def linear_search(arr, x):
2     for i in range(len(arr)):
3         if arr[i] == x:
4             return i
5     return -1
```

## 4. Bài tập

### Bài 1: Tìm chỉ số đầu tiên

Viết hàm `linear_search(arr, x)` trả về chỉ số đầu tiên mà `x` xuất hiện trong `arr`, hoặc `-1` nếu không tìm thấy.

**Ví dụ:**

- arr = [2, 4, 6, 8, 10], x = 6 → Output: 2
- arr = [1, 3, 5, 7], x = 2 → Output: -1

## Bài 2: Tìm tất cả các chỉ số

Viết hàm trả về danh sách tất cả chỉ số mà x xuất hiện trong arr.

Ví dụ: arr = [1, 2, 3, 2, 4, 2], x = 2 → Output: [1, 3, 5]

## Bài 3: Tìm phần tử lớn hơn

Viết hàm find\_greater(arr, x) trả về danh sách các phần tử lớn hơn x trong arr.

## Bài 4: Tìm phần tử đầu tiên chia hết cho 3

Viết hàm find\_first\_div\_by\_3(arr) trả về phần tử đầu tiên chia hết cho 3 trong danh sách.

# Thuật Toán Bubble Sort

*Hoàng-Nguyễn Vũ*

## 1. Khái niệm

**Bubble Sort** là thuật toán sắp xếp đơn giản hoạt động bằng cách so sánh các cặp phần tử liền kề và hoán đổi nếu chúng không theo đúng thứ tự mong muốn (ví dụ: tăng dần). Quá trình này được lặp lại cho đến khi mảng được sắp xếp hoàn toàn.

## 2. Nguyên lý hoạt động

- Lặp qua mảng nhiều lần.
- Trong mỗi vòng lặp, so sánh hai phần tử kề nhau:
  - Nếu  $\text{arr}[i] > \text{arr}[i+1]$  thì hoán đổi.
- Sau mỗi vòng, phần tử lớn nhất sẽ "nổi" về cuối mảng.

## 3. Độ phức tạp

- Trường hợp tốt nhất:  $O(n)$  (nếu dùng flag kiểm tra đã sắp xếp)
- Trung bình và xấu nhất:  $O(n^2)$
- Không sử dụng bộ nhớ phụ:  $O(1)$

## 4. Giả mã (Pseudocode)

```
1 for i in range(0, n-1):
2     for j in range(0, n-i-1):
3         if arr[j] > arr[j+1]:
4             swap arr[j] and arr[j+1]
```

## 5. Cài đặt Python

```
1 def bubble_sort(arr):
2     n = len(arr)
3     for i in range(n):
4         swapped = False
5         for j in range(0, n - i - 1):
6             if arr[j] > arr[j+1]:
```

```
7         arr[j], arr[j+1] = arr[j+1], arr[j]
8         swapped = True
9     if not swapped:
10        break
```

## 6. Bài tập thực hành

### Bài tập 1: Sắp xếp tăng dần

Viết hàm bubble\_sort(arr) để sắp xếp mảng theo thứ tự tăng dần.

Test:

- arr = [5, 3, 8, 4, 2] ⇒ Output: [2, 3, 4, 5, 8]

### Bài tập 2: Sắp xếp giảm dần

Viết lại thuật toán Bubble Sort để sắp xếp mảng theo thứ tự giảm dần.

### Bài tập 3: Đếm số lần hoán đổi

Viết hàm bubble\_sort\_count(arr) sắp xếp mảng và trả về số lần hoán đổi.

Test:

- arr = [4, 3, 2, 1] ⇒ Output: 6

### Bài tập 4: Kiểm tra đã sắp xếp

Trước khi thực hiện sắp xếp, kiểm tra nếu mảng đã được sắp thì in "Already sorted" và không thực hiện hoán đổi.

# Thuật Toán Binary Search (Tìm kiếm nhị phân)

*Hoàng-Nguyễn Vũ*

## 1. Khái niệm

**Binary Search** là thuật toán tìm kiếm nhị phân, giúp tìm kiếm phần tử trong mảng đã sắp xếp một cách hiệu quả. Thuật toán hoạt động bằng cách chia đôi khoảng tìm kiếm trong mỗi bước so sánh.

## 2. Điều kiện áp dụng

- Mảng phải được **sắp xếp**.
- Có thể triển khai bằng **vòng lặp hoặc đệ quy**.

## 3. Độ phức tạp

- Trường hợp tốt nhất:  $O(1)$
- Trung bình và xấu nhất:  $O(\log n)$
- Không gian:  $O(1)$  nếu dùng vòng lặp,  $O(\log n)$  nếu đệ quy

## 4. Giả mã

```
1 1. Đặt left = 0, right = n - 1
2 2. Trong khi left <= right:
3   a. mid = (left + right) // 2
4   b. Nếu arr[mid] == x      trả về mid
5   c. Nếu arr[mid] < x      tìm bên phải (left = mid + 1)
6   d. Nếu arr[mid] > x      tìm bên trái (right = mid - 1)
7 3. Nếu không tìm thấy      trả về -1
```

## 5. Cài đặt Python (vòng lặp)

```
1 def binary_search(arr, x):
2     left = 0
3     right = len(arr) - 1
4     while left <= right:
5         mid = (left + right) // 2
6         if arr[mid] == x:
7             return mid
```

```
8     elif arr[mid] < x:
9         left = mid + 1
10    else:
11        right = mid - 1
12    return -1
```

## 6. Bài tập thực hành

### Bài 1: Tìm vị trí phần tử

Viết hàm `binary_search(arr, x)` trả về chỉ số của `x` nếu có, ngược lại trả về `-1`.

Ví dụ:

- `arr = [1, 3, 5, 7, 9]`, `x = 5` → Output: 2
- `x = 2` → Output: -1

### Bài 2: Tìm phần tử đầu tiên xuất hiện

Viết hàm tìm chỉ số đầu tiên của phần tử `x` trong mảng có thể chứa nhiều phần tử giống nhau.

Ví dụ: `arr = [1, 2, 4, 4, 4, 5]`, `x = 4` → Output: 2

### Bài 3: Tìm phần tử nhỏ nhất $\geq x$

Trả về chỉ số của phần tử đầu tiên trong mảng lớn hơn hoặc bằng `x`.

Ví dụ:

- `arr = [1, 3, 5, 7, 9]`, `x = 6` → Output: 3
- `x = 10` → Output: -1

### Bài 4: Đếm số lần gọi đệ quy

Viết lại Binary Search dưới dạng đệ quy, và trả về số lần gọi hàm tìm kiếm.

# Bài Tập Ôn Tập Thuật Toán và Ứng Dụng AI

*Hoàng-Nguyễn Vũ*

## 1. Linear Search – Ứng dụng AI

### Bài 1.1 – Tìm prompt chứa từ khoá

```

1 def find_prompt_with_keyword(prompts, keyword):
2     ## Your code here ##
3
4 prompts = [
5     "Generate a report for patient X.",
6     "Summarize the input text.",
7     "Suggest a diagnosis based on symptoms.",
8     "Translate this to French."
9 ]
10 print(find_prompt_with_keyword(prompts, "diagnosis"))
11 # Output: 2

```

### Bài 1.2 – Tìm mẫu ảnh chưa có annotation

```

1 def find_first_unlabeled(images):
2     ## Your code here ##
3
4 images = [
5     {'name': 'img1.png', 'label': 'benign'},
6     {'name': 'img2.png', 'label': None},
7     {'name': 'img3.png', 'label': 'malignant'}
8 ]
9 print(find_first_unlabeled(images))
10 # Output: 1

```

## 2. Bubble Sort – Ứng dụng AI

### Bài 2.1 – Sắp xếp ảnh theo độ mờ

```

1 def sort_by_blurriness(images):
2     ## Your code here ##
3
4 image_data = [
5     {'name': 'img1', 'blurriness': 0.8},
6     {'name': 'img2', 'blurriness': 0.3},
7     {'name': 'img3', 'blurriness': 0.5}
8 ]
9 sorted_imgs = sort_by_blurriness(image_data)

```

```

10 print([img['name'] for img in sorted_imgs])
11 # Output: ['img2', 'img3', 'img1']

```

## Bài 2.2 – Sắp xếp mô hình theo độ chính xác

```

1 def sort_models_by_accuracy(models, accuracies):
2     ## Your code here ##
3
4 models = ["ModelA", "ModelB", "ModelC"]
5 accuracies = [0.89, 0.93, 0.91]
6 print(sort_models_by_accuracy(models, accuracies))
7 # Output: ['ModelB', 'ModelC', 'ModelA']

```

## Bài 2.3 – Đếm số lần hoán đổi theo loss

```

1 def bubble_sort_with_count(arr):
2     count = 0
3     ## Your code here ##
4     return count
5
6 losses = [0.4, 0.3, 0.5, 0.2]
7 print(bubble_sort_with_count(losses))
8 # Output: 4

```

## 3. Binary Search – Ứng dụng AI

### Bài 3.1 – Tìm xác suất bệnh $\geq$ ngưỡng

```

1 def find_first_greater_or_equal(arr, x):
2     left, right = 0, len(arr) - 1
3     result = -1
4     ## Your code here ##
5     return result
6
7 probs = [0.12, 0.23, 0.35, 0.47, 0.59, 0.61, 0.74, 0.81]
8 print(find_first_greater_or_equal(probs, 0.6))
9 # Output: 5

```

### Bài 3.2 – Tìm threshold có F1-score tốt nhất

```

1 def binary_search_threshold(f1_scores, threshold):
2     left, right = 0, len(f1_scores) - 1
3     ## Your code here ##
4     return None
5
6 f1_scores = [(0.1, 0.6), (0.2, 0.65), (0.3, 0.7), (0.4, 0.68)]
7 print(binary_search_threshold(f1_scores, 0.3))
8 # Output: 0.7

```

### Bài 3.3 – Kiểm tra có ảnh tương đồng cao không

```
1 def exists_similar_image(similarities, threshold):
2     left, right = 0, len(similarities) - 1
3     ## Your code here ##
4     return False
5
6 sims = [0.95, 0.89, 0.85, 0.80, 0.75] # descending order
7 print(exists_similar_image(sims, 0.85))
8 # Output: True
```

# AI Dataset Quality Checker – Áp dụng Linear Search, Bubble Sort và Binary Search

*Hoàng-Nguyễn Vũ*

**Mục tiêu:** Xây dựng chương trình kiểm tra chất lượng dữ liệu ảnh trong quá trình chuẩn bị tập huấn luyện mô hình AI.

## Dữ liệu mẫu

```
1 dataset = [
2     {"name": "img1.png", "label": "benign", "blurriness": 0.4, "similarity": 0.92},
3     {"name": "img2.png", "label": None, "blurriness": 0.6, "similarity": 0.95},
4     {"name": "img3.png", "label": "malignant", "blurriness": 0.3, "similarity": 0.84},
5     {"name": "img4.png", "label": None, "blurriness": 0.5, "similarity": 0.88}
6 ]
```

## Yêu cầu chức năng

**Bước 1. Tìm ảnh chưa được gán nhãn:** Dùng Linear Search để xác định các ảnh chưa có label.

**Bước 2. Sắp xếp ảnh theo độ mờ:** Dùng Bubble Sort để sắp xếp ảnh theo trường `blurriness` tăng dần.

**Bước 3. Phát hiện ảnh tương tự:** Dùng Binary Search để kiểm tra xem có ảnh nào có `similarity > 0.9` không.

## Mở rộng

- Ghi log ảnh không đạt chất lượng ra file.
- Thêm kiểm tra ảnh trùng (dựa theo độ tương đồng cao và tên tệp gần giống).
- Tạo báo cáo tổng hợp số lượng ảnh tốt, ảnh mờ, ảnh chưa gán nhãn.

# Làm quen với FastAPI

*Hoàng-Nguyễn Vũ*

## 1. Lý thuyết cơ bản về FastAPI

### FastAPI là gì?

FastAPI là một framework hiện đại, nhanh (hiệu năng gần ngang NodeJS và Go) dùng để xây dựng RESTful API với Python 3.7+, dựa trên:

- Type hints
- ASGI (Asynchronous Server Gateway Interface)

### Lợi ích chính

- Rất nhanh và hiệu quả
- Tự động sinh tài liệu API (Swagger UI, ReDoc)
- Hỗ trợ async/await
- Xác thực và kiểm tra dữ liệu đầu vào mạnh mẽ bằng Pydantic

### Cài đặt

```
1 pip install fastapi uvicorn
```

### Ví dụ Hello API

```
1 from fastapi import FastAPI
2
3 app = FastAPI()
4
5 @app.get("/")
6 def read_root():
7     return {"message": "Hello World"}
```

Chạy server:

```
1 uvicorn main:app --reload
```

### API với tham số URL

```
1 @app.get("/items/{item_id}")
2 def read_item(item_id: int, q: str = None):
3     return {"item_id": item_id, "q": q}
```

## POST API sử dụng Pydantic

```
1 from pydantic import BaseModel
2
3 class Item(BaseModel):
4     name: str
5     price: float
6     is_offer: bool = False
7
8 @app.post("/items/")
9 def create_item(item: Item):
10     return {"received": item}
```

## 2. Bài tập thực hành

**Bài 1. Hello API:** Tạo một route GET trả về chuỗi "Xin chào từ FastAPI!" tại /hello.

**Bài 2. API Máy tính:** Tạo 4 route GET như sau:

- /add?a=5&b=3 → trả về 8
- /subtract?a=10&b=7 → trả về 3
- /multiply?a=4&b=2 → trả về 8
- /divide?a=8&b=2 → trả về 4 (xử lý chia cho 0)

**Bài 3. API Thông tin người dùng:** Tạo một API POST /user nhận dữ liệu JSON gồm:

- name (str)
- age (int)
- email (str)

Và trả về: is\_adult = True nếu tuổi  $\geq 18$ .

**Bài 4. Quản lý danh sách sản phẩm:**

- GET /products trả về danh sách sản phẩm
- POST /products thêm sản phẩm vào danh sách (lưu tạm bằng list)

## 3. Tài liệu tham khảo

- <https://fastapi.tiangolo.com/>
- <https://github.com/tiangolo/fastapi>

# Thiết kế cơ sở dữ liệu cho hệ thống hồ sơ bệnh án điện tử (EMR)

*Hoàng-Nguyễn Vũ*

## 1. Đặt vấn đề

Hồ sơ bệnh án điện tử (Electronic Medical Record – EMR) là thành phần trọng yếu trong hệ sinh thái số hóa y tế, đóng vai trò trung tâm trong việc lưu trữ, quản lý và truy xuất thông tin bệnh nhân một cách hiệu quả, nhất quán và an toàn. Để xây dựng được một hệ thống EMR hiệu quả, bước khởi đầu mang tính nền tảng là phân tích nghiệp vụ và thiết kế cơ sở dữ liệu (CSDL) hợp lý, có khả năng mô hình hóa chính xác các thực thể và mối liên hệ trong bối cảnh lâm sàng.

Tài liệu này trình bày các nguyên lý phân tích và thiết kế CSDL cho hệ thống EMR dưới góc nhìn học thuật, kết hợp lý thuyết mô hình hóa dữ liệu với đặc thù của ngành y tế, nhằm phục vụ mục tiêu phát triển hệ thống thông tin y tế bền vững và có khả năng mở rộng.

## 2. Phân tích yêu cầu dữ liệu

Trước khi tiến hành thiết kế dữ liệu, cần nhận diện rõ các thành phần thông tin thiết yếu trong một hệ thống EMR, bao gồm:

- Thông tin định danh bệnh nhân: họ tên, ngày sinh, giới tính, địa chỉ, mã số định danh y tế.
- Dữ liệu lâm sàng và cận lâm sàng: các triệu chứng, chẩn đoán, kết quả xét nghiệm, hình ảnh học.
- Hoạt động điều trị: đơn thuốc, phác đồ điều trị, thủ thuật và theo dõi tiến triển.
- Thông tin hành chính – tổ chức: bác sĩ phụ trách, thời gian tiếp nhận, cơ sở y tế, đơn vị chuyên khoa.

Việc phân tích yêu cầu dữ liệu cần đặt trong ngữ cảnh thực hành lâm sàng, bảo đảm rằng mô hình dữ liệu không chỉ phù hợp với lý thuyết cơ sở dữ liệu, mà còn đáp ứng đầy đủ các nhu cầu khai thác thông tin thực tế tại các cơ sở khám chữa bệnh.

## 3. Nhận diện các thực thể và thuộc tính

Dựa trên các yêu cầu đã phân tích, có thể xác định một số thực thể cốt lõi trong mô hình dữ liệu EMR như sau:

- Patient (Bệnh nhân):** đại diện cho đối tượng sử dụng dịch vụ y tế, chứa các thuộc tính liên quan đến nhân khẩu học và hành chính.

2. **Visit (Lượt khám):** biểu diễn mỗi lần tương tác giữa bệnh nhân và cơ sở y tế. Một bệnh nhân có thể có nhiều lượt khám tại các thời điểm khác nhau.
3. **Prescription (Đơn thuốc):** lưu trữ thông tin về thuốc được kê cho bệnh nhân trong mỗi lượt khám, bao gồm tên thuốc, liều lượng, tần suất dùng,...
4. **Test (Xét nghiệm):** phản ánh các chỉ định cận lâm sàng được yêu cầu trong một lượt khám, cùng với kết quả và thời gian thực hiện.

Các thực thể trên không tồn tại độc lập mà có quan hệ liên kết theo các ràng buộc nghiệp vụ cụ thể, đòi hỏi mô hình dữ liệu phải thể hiện đầy đủ và chính xác các mối quan hệ này.

## 4. Mô hình hóa quan hệ giữa các thực thể

Quan hệ giữa các thực thể trong hệ thống EMR có thể được mô hình hóa theo sơ đồ thực thể – liên kết (Entity-Relationship Diagram – ERD). Một số quan hệ tiêu biểu:

- Quan hệ 1 – N giữa bệnh nhân và lượt khám: một bệnh nhân có thể có nhiều lượt khám khác nhau.
- Quan hệ 1 – N giữa lượt khám và đơn thuốc: mỗi lượt khám có thể phát sinh nhiều đơn thuốc.
- Quan hệ 1 – N giữa lượt khám và xét nghiệm: mỗi lượt khám có thể gắn liền với nhiều xét nghiệm.

Các quan hệ cần được biểu diễn tường minh thông qua cơ chế khóa chính – khóa ngoại nhằm đảm bảo tính toàn vẹn tham chiếu và hỗ trợ truy vấn hiệu quả.

## 5. Chuyển đổi mô hình ER sang lược đồ quan hệ

Việc chuyển đổi mô hình ER thành lược đồ CSDL quan hệ cần thực hiện theo các bước sau:

1. Gán mỗi thực thể thành một bảng (relation), xác định rõ khóa chính.
2. Với mỗi quan hệ 1 – N, thêm khóa ngoại vào bảng phía N.
3. Với các thuộc tính lặp lại, cần xem xét chuẩn hóa dữ liệu nhằm loại bỏ dư thừa và tránh mâu thuẫn.

Việc chuẩn hóa dữ liệu nên được thực hiện đến ít nhất là dạng chuẩn thứ ba (3NF) để bảo đảm tính đơn trị, tính phụ thuộc hàm đầy đủ, và loại bỏ các phụ thuộc bắc cầu.

## 6. Đề xuất hoạt động tự đánh giá và ứng dụng

### Bài tập phân tích – mô hình hóa

Phân tích một hệ thống quản lý bệnh nhân ngoại trú tại bệnh viện tuyến tỉnh, xác định:

- Danh sách các thực thể và thuộc tính tương ứng.
- Các mối quan hệ giữa thực thể, xác định kiểu quan hệ và ràng buộc.
- Vẽ sơ đồ ER thể hiện rõ các mối liên kết dữ liệu.

### Bài tập thiết kế cơ sở dữ liệu quan hệ

Chuyển đổi sơ đồ ER ở trên thành tập lược đồ bảng đầy đủ, chỉ định:

- Tên bảng, danh sách cột và kiểu dữ liệu.
- Khóa chính và khóa ngoại của từng bảng.
- Nhận xét về mức độ chuẩn hóa của thiết kế.

### Bài tập đánh giá mô hình dữ liệu

Phân tích ưu – nhược điểm của mô hình dữ liệu đã thiết kế theo các tiêu chí:

- Hiệu quả lưu trữ
- Tính toàn vẹn dữ liệu
- Khả năng mở rộng
- Dễ dàng truy vấn phân tích

# Thiết kế và triển khai giao diện lập trình ứng dụng (API) cho hệ thống hồ sơ bệnh án điện tử (EMR) sử dụng FastAPI

Hoàng-Nguyễn Vũ

## 1. Mở đầu

Việc tích hợp cơ sở dữ liệu bệnh án với một giao diện lập trình ứng dụng (API) là bước quan trọng nhằm hỗ trợ truy cập, xử lý và chia sẻ dữ liệu y tế một cách hiệu quả giữa các hệ thống thông tin. FastAPI là một framework hiện đại, hiệu năng cao trong hệ sinh thái Python, đặc biệt thích hợp cho việc xây dựng RESTful API nhờ khả năng hỗ trợ typing, sinh tài liệu tự động và xử lý bất đồng bộ.

Tài liệu này trình bày nguyên lý tích hợp cơ sở dữ liệu EMR vào một Web API sử dụng FastAPI, bao gồm lý thuyết thiết kế và các bài tập ứng dụng có định hướng nghiên cứu và phát triển.

## 2. Mục tiêu triển khai API

- Tạo giao diện truy xuất dữ liệu hồ sơ bệnh án theo mô hình REST.
- Phân tách rõ lớp xử lý dữ liệu (database) và lớp dịch vụ (service/API).
- Hỗ trợ các thao tác CRUD (Create, Read, Update, Delete) đối với từng thực thể dữ liệu trong hệ thống EMR.
- Tạo tài liệu tương tác tự động (Swagger UI) nhằm hỗ trợ thử nghiệm và phát triển mở rộng.

## 3. Kiến trúc tích hợp cơ bản

Mô hình kiến trúc API cho EMR theo hướng phân lớp bao gồm:

- Lớp mô hình dữ liệu (Data Models):** ánh xạ các bảng trong cơ sở dữ liệu vào các lớp Python thông qua Pydantic.
- Lớp truy xuất dữ liệu (Database Layer):** thực hiện các truy vấn SQL tương tác với hệ quản trị cơ sở dữ liệu (MySQL/PostgreSQL).
- Lớp API (Router):** định nghĩa các endpoint RESTful phục vụ truy cập từ phía client.

Việc tổ chức mã nguồn cần tuân theo nguyên tắc tách biệt mối quan tâm (Separation of Concerns – SoC) nhằm tăng khả năng bảo trì, kiểm thử và mở rộng.

## 4. Gợi ý các tài nguyên API cho EMR

### 1. /patients:

- GET: Truy xuất danh sách bệnh nhân
- POST: Tạo hồ sơ bệnh nhân mới

### 2. /patients/{id}:

- GET: Xem thông tin một bệnh nhân
- PUT: Cập nhật thông tin
- DELETE: Xóa hồ sơ bệnh nhân

### 3. /visits:

- POST: Ghi nhận một lượt khám mới

### 4. /patients/{id}/history:

- GET: Truy xuất toàn bộ lịch sử khám bệnh, đơn thuốc và xét nghiệm của một bệnh nhân

## 5. Bài tập ứng dụng (theo hướng phát triển hệ thống)

### Bài 1: Khởi tạo ứng dụng FastAPI

Khởi tạo dự án FastAPI cho hệ thống EMR, bao gồm:

- Cấu trúc thư mục rõ ràng: app/main.py, app/routers/, app/models/, app/database/.
- Khai báo cấu hình kết nối cơ sở dữ liệu.
- Tạo endpoint kiểm tra: /ping trả về "pong".

### Bài 2: Xây dựng endpoint cho quản lý bệnh nhân

Thiết kế các route API cho tài nguyên /patients, thực hiện các thao tác:

- Tạo mới hồ sơ bệnh nhân
- Truy vấn tất cả bệnh nhân
- Truy vấn bệnh nhân theo ID
- Cập nhật và xóa bệnh nhân

### Bài 3: Quản lý lượt khám bệnh

Mở rộng API với các chức năng sau:

- Ghi nhận một lượt khám mới cho bệnh nhân
- Tìm danh sách lượt khám theo ID bệnh nhân
- Lọc lượt khám theo ngày hoặc bác sĩ phụ trách

### Bài 4: Triển khai chức năng truy xuất bệnh án đầy đủ

Thiết kế một endpoint `/patients/{id}/history` có khả năng:

- Truy xuất toàn bộ lượt khám của bệnh nhân
- Bao gồm các đơn thuốc và xét nghiệm liên quan trong từng lượt khám
- Kết xuất kết quả dưới dạng JSON có cấu trúc phân tầng rõ ràng

### Bài 5: Tài liệu hóa và thử nghiệm API

Thực hiện kiểm tra toàn bộ hệ thống API thông qua giao diện tài liệu:

- Truy cập Swagger UI tại `/docs`
- Kiểm tra phản hồi dữ liệu với các trường hợp hợp lệ và bất hợp lệ
- Ghi nhận các mã trạng thái HTTP trả về (200, 201, 404, 422,...)

# Truy xuất thông tin bệnh nhân bằng Function Calling trong LLM và tích hợp với hệ thống API y tế

Hoàng-Nguyễn Vũ

## 1. Bối cảnh và mục tiêu

Trong các hệ thống hỗ trợ ra quyết định y tế thông minh, khả năng truy xuất thông tin bệnh nhân bằng ngôn ngữ tự nhiên đóng vai trò quan trọng trong việc nâng cao hiệu suất tương tác người-máy. Sự kết hợp giữa mô hình ngôn ngữ lớn (LLM) và giao diện lập trình ứng dụng (API) thông qua kỹ thuật Function Calling mở ra hướng tiếp cận hiệu quả để giải quyết bài toán này.

Tài liệu này trình bày lý thuyết cốt lõi và thiết kế bài tập thực hành xoay quanh việc sử dụng Function Calling trong ChatGPT-4o để thực hiện truy vấn dữ liệu bệnh nhân từ hệ thống API y tế xây dựng bằng FastAPI.

## 2. Khái niệm Function Calling trong ngữ cảnh LLM

Function Calling là cơ chế cho phép LLM không chỉ phản hồi dưới dạng văn bản, mà còn sinh ra lời gọi hàm (function call) có cấu trúc, thường là đối tượng JSON, dựa trên schema do nhà phát triển cung cấp. Khi được kết nối với một hệ thống backend, lời gọi này có thể được thực thi để truy xuất thông tin, từ đó trả kết quả về cho người dùng thông qua LLM.

## 3. Bài toán truy xuất thông tin bệnh nhân (Patient Retrieval)

### 3.1. Đặc tả bài toán

Cho phép người dùng đặt câu hỏi tự nhiên nhằm tìm kiếm thông tin bệnh nhân dựa trên các tiêu chí như:

- Họ tên bệnh nhân
- Mã số bệnh nhân (ID)
- Các đặc trưng khác: giới tính, năm sinh (tùy mở rộng)

### 3.2. Luồng hoạt động

1. Người dùng nhập câu hỏi như: “Tìm bệnh nhân tên Lê Minh Đức”
2. Mô hình ChatGPT nhận diện ý định và sinh function call dưới dạng JSON.

3. Hệ thống backend nhận lệnh và truy vấn API (FastAPI).
4. Kết quả được xử lý và phản hồi lại cho người dùng bằng ngôn ngữ tự nhiên.

## 4. Định nghĩa Function Schema cho truy vấn bệnh nhân

Tên hàm:

retrieve\_patient

Mô tả:

Truy xuất thông tin bệnh nhân từ hệ thống hồ sơ dựa trên tên hoặc ID.

### Định nghĩa schema (OpenAI function calling)

```
1 {
2     "name": "retrieve_patient",
3     "description": "Truy xuất thông tin bệnh nhân theo tên hoặc ID.",
4     "parameters": {
5         "type": "object",
6         "properties": {
7             "id": {
8                 "type": "integer",
9                 "description": "ID bệnh nhân nếu biết rõ"
10            },
11            "name": {
12                "type": "string",
13                "description": "Họ tên bệnh nhân cần truy xuất"
14            }
15        },
16        "required": []
17    }
18 }
```

## 5. Thiết kế bài tập thực hành

### Bài 1: Phân tích truy vấn người dùng

Phân tích câu lệnh sau: “Tôi muốn tìm thông tin bệnh nhân tên là Nguyễn Thị Hồng”

Yêu cầu:

- Xác định hàm được gọi
- Sinh ra JSON đầu vào phù hợp với schema đã định nghĩa

Gợi ý kết quả mong đợi:

```
1 {  
2   "name": "Nguyễn Thị Hồng"  
3 }
```

## Bài 2: Tích hợp với FastAPI

Xây dựng endpoint tương ứng với function `retrieve_patient` sử dụng FastAPI, cho phép:

- Tìm bệnh nhân theo `name` hoặc `id`
- Trả về kết quả ở dạng JSON

**Yêu cầu mở rộng:** xử lý trường hợp không tìm thấy bệnh nhân.

## Bài 3: Sinh function call từ nhiều cách diễn đạt

Thử các câu hỏi với các cấu trúc ngữ nghĩa khác nhau:

- “Ai là người có ID 102?”
- “Tôi cần thông tin bệnh nhân tên Trần Văn Bình”
- “Cho tôi xem hồ sơ của bệnh nhân có mã 456”

Dánh giá độ chính xác khi mô hình sinh ra JSON function call.

## Bài 4: Xây dựng phản hồi tự nhiên từ dữ liệu API

Cho trước kết quả API:

```
{  
  "id": 102,  
  "name": "Trần Văn Bình",  
  "dob": "1990-02-01",  
  "gender": "Male"
```

Viết đoạn phản hồi tự nhiên (natural language) mô hình có thể sinh ra dựa trên dữ liệu này.

**Gợi ý:** “Bệnh nhân Trần Văn Bình, sinh ngày 01/02/1990, giới tính Nam, có mã số bệnh nhân là 102.”

# Các Phương Pháp Tính Khoảng Cách Giữa Vector

*Hoàng-Nguyễn Vũ*

## 1. Lý thuyết

### 1.1 Euclidean Distance (Khoảng cách Euclid)

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

**Ý nghĩa:** Khoảng cách đường thẳng trong không gian  $n$  chiều.

**Ứng dụng:** KNN, Clustering, PCA, v.v.

### 1.2 Manhattan Distance (Khoảng cách Manhattan)

$$d(x, y) = \sum_{i=1}^n |x_i - y_i|$$

**Ý nghĩa:** Khoảng cách tổng theo từng trục tọa độ.

**Ứng dụng:** Dữ liệu dạng lưới, chông nhiễu tốt hơn Euclidean.

### 1.3 Cosine Similarity / Distance

$$\cos(x, y) = \frac{x \cdot y}{\|x\| \|y\|} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum x_i^2} \cdot \sqrt{\sum y_i^2}}$$

$$\text{Cosine Distance} = 1 - \cos(x, y)$$

**Ý nghĩa:** So sánh góc giữa hai vector, bất kể độ dài.

**Ứng dụng:** NLP, phân tích văn bản, vector hóa embedding.

### 1.4 Minkowski Distance

$$d(x, y) = \left( \sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}$$

**Đặc biệt:**

- $p = 1 \rightarrow$  Manhattan
- $p = 2 \rightarrow$  Euclidean
- $p \rightarrow \infty \rightarrow$  Chebyshev

## 1.5 Chebyshev Distance

$$d(x, y) = \max_i |x_i - y_i|$$

**Ý nghĩa:** Khoảng cách lớn nhất theo bất kỳ chiều nào.

**Ứng dụng:** Điều hướng theo ô vuông, game, robot.

## 2. Bài tập Áp Dụng

Hãy viết các function trong Python, cài đặt các phương pháp tính khoảng cách.

### Bài 1: Euclidean và Manhattan Distance

Cho hai vector:

$$x = [3, 4], \quad y = [0, 0]$$

Tính:

- Euclidean distance
- Manhattan distance

**Đáp án:**

$$\text{Euclidean} = \sqrt{(3 - 0)^2 + (4 - 0)^2} = \sqrt{25} = 5$$

$$\text{Manhattan} = |3 - 0| + |4 - 0| = 7$$

### Bài 2: Cosine Similarity và Distance

Cho:

$$x = [1, 2], \quad y = [2, 4]$$

Tính:

- Cosine similarity
- Cosine distance

**Gợi ý:**

$$x \cdot y = 1 \cdot 2 + 2 \cdot 4 = 10$$

$$\|x\| = \sqrt{1^2 + 2^2} = \sqrt{5}, \quad \|y\| = \sqrt{2^2 + 4^2} = \sqrt{20}$$

$$\text{Cosine similarity} = \frac{10}{\sqrt{5} \cdot \sqrt{20}} = 1 \quad \Rightarrow \quad \text{Cosine distance} = 0$$

### Bài 3: So sánh các loại khoảng cách

Cho:

$$x = [5, 8, 2], \quad y = [3, 5, 4]$$

Tính:

- Euclidean distance
- Manhattan distance
- Chebyshev distance

Gợi ý:

$$\text{Euclidean} = \sqrt{(5 - 3)^2 + (8 - 5)^2 + (2 - 4)^2} = \sqrt{4 + 9 + 4} = \sqrt{17}$$

$$\text{Manhattan} = |5 - 3| + |8 - 5| + |2 - 4| = 2 + 3 + 2 = 7$$

$$\text{Chebyshev} = \max(|5 - 3|, |8 - 5|, |2 - 4|) = \max(2, 3, 2) = 3$$

# Thuật Toán K-Nearest Neighbors (KNN)

*Hoàng-Nguyễn Vũ*

## 1. Lý Thuyết

### 1.1 Định nghĩa

K-Nearest Neighbors (KNN) là một thuật toán học máy dạng **không tham số** (non-parametric) và thuộc nhóm **học có giám sát** (supervised learning).

Ý tưởng chính của KNN là:

Một điểm dữ liệu sẽ được gán nhãn theo **đa số nhãn** của  $K$  điểm gần nhất trong tập huấn luyện.

### 1.2 Các bước của thuật toán KNN

1. Xác định số lượng hàng xóm gần nhất  $K$ .
2. Tính khoảng cách giữa điểm cần phân loại và tất cả các điểm trong tập huấn luyện (thường dùng **Euclidean distance**).
3. Chọn ra  $K$  điểm gần nhất.
4. Lấy nhãn phổ biến nhất trong  $K$  điểm đó để gán cho điểm mới.

### 1.3 Các loại khoảng cách thường dùng

- **Euclidean distance:**  $d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$
- **Manhattan distance:**  $d(x, y) = \sum_{i=1}^n |x_i - y_i|$
- **Minkowski distance:**  $d(x, y) = (\sum_{i=1}^n |x_i - y_i|^p)^{1/p}$

### 1.4 Ưu điểm và Nhược điểm

#### Ưu điểm:

- Dễ hiểu, dễ cài đặt.
- Không cần huấn luyện mô hình (lazy learning).

#### Nhược điểm:

- Tính toán chậm với tập dữ liệu lớn.
- Nhạy cảm với outliers và feature scale.

## 2. Bài Tập Áp Dụng

### Bài 1: Tính khoảng cách

Cho tập dữ liệu huấn luyện như sau:

Điểm	x	y	Nhãn
A	1	2	0
B	2	3	0
C	3	1	1
D	6	5	1

Hãy tính **Euclidean distance** từ điểm  $P = (2, 2)$  đến tất cả các điểm trong tập và sắp xếp theo thứ tự tăng dần.

### Bài 2: Phân loại với KNN

Sử dụng dữ liệu từ Bài 1, hãy phân loại điểm  $P = (2, 2)$  với:

- $K = 1$
- $K = 3$

Gợi ý:

- Tìm  $K$  điểm gần nhất theo khoảng cách Euclid.
- Đếm nhãn của các điểm gần nhất.
- Chọn nhãn phổ biến nhất.

### Bài 3: KNN trong không gian 3 chiều

Cho:

$$x = (2, 1, 4), \quad y = (5, 3, 0)$$

Tính khoảng cách Euclid giữa  $x$  và  $y$ .

$$d(x, y) = ?$$

### Bài 4: Ảnh hưởng của K đến kết quả

Giả sử bạn có tập dữ liệu gồm 9 điểm, trong đó:

- 5 điểm có nhãn A
- 4 điểm có nhãn B

Nếu chọn:

- $K = 3$
- $K = 5$
- $K = 9$

Trong trường hợp điểm cần phân loại có các hàng xóm gần nhất được chọn từ nhóm 9 điểm trên, hãy cho biết nhãn sẽ được gán theo từng giá trị  $K$ .

### 3. Gợi ý Cài đặt Python (Pseudocode)

```
1 from sklearn.neighbors import KNeighborsClassifier
2
3 # Dữ liệu huấn luyện
4 X_train = [[1, 2], [2, 3], [3, 1], [6, 5]]
5 y_train = [0, 0, 1, 1]
6
7 # Dự đoán điểm mới
8 model = KNeighborsClassifier(n_neighbors=3)
9 model.fit(X_train, y_train)
10
11 print(model.predict([[2, 2]]))
```

# Thuật Toán K-Nearest Neighbors (KNN) - Nâng cao

*Hoàng-Nguyễn Vũ*

## 1. Lý Thuyết

### 1.1 Định nghĩa

K-Nearest Neighbors (KNN) là một thuật toán học máy dạng **không tham số** (non-parametric) và thuộc nhóm **học có giám sát** (supervised learning).

Ý tưởng chính của KNN là:

Một điểm dữ liệu sẽ được gán nhãn theo **đa số nhãn** của  $K$  điểm gần nhất trong tập huấn luyện.

### 1.2 Các bước của thuật toán KNN

1. Xác định số lượng hàng xóm gần nhất  $K$ .
2. Tính khoảng cách giữa điểm cần phân loại và tất cả các điểm trong tập huấn luyện (thường dùng **Euclidean distance**).
3. Chọn ra  $K$  điểm gần nhất.
4. Lấy nhãn phổ biến nhất trong  $K$  điểm đó để gán cho điểm mới.

### 1.3 Các loại khoảng cách thường dùng

- **Euclidean distance:**  $d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$
- **Manhattan distance:**  $d(x, y) = \sum_{i=1}^n |x_i - y_i|$
- **Minkowski distance:**  $d(x, y) = (\sum_{i=1}^n |x_i - y_i|^p)^{1/p}$

## 2. Bài tập và Code Python Không Dùng Thư Viện

### Bài 1: Viết tay KNN phân loại nhị phân

**Yêu cầu:** Viết hàm dự đoán nhãn nhị phân dựa trên khoảng cách Euclidean.

```

1 import math
2 from collections import Counter
3
4 def euclidean_distance(x1, x2):
5     return # Your code here #
6
7 def knn_predict(X_train, y_train, x_test, k=3):

```

```
8     distances = []
9     # Your code here #
10    return # Your code here #
11
12 # Example
13 X_train = [[1, 2], [2, 3], [3, 1], [6, 5]]
14 y_train = [0, 0, 1, 1]
15 x_test = [2, 2]
16 print("Predicted label:", knn_predict(X_train, y_train, x_test, k=3))
```

## Bài 2: Viết KNN hồi quy (Regression)

```
1 def knn_regression(X_train, y_train, x_test, k=3):
2     distances = []
3     # Your code here #
4     return # Your code here #
5
6 # Dự đoán giá nhà
7 X_train = [[100], [150], [200], [250], [300]]
8 y_train = [1.0, 1.5, 2.0, 2.5, 3.0]
9 x_test = [220]
10 print("Predicted price:", knn_regression(X_train, y_train, x_test, k=2))
```

## Bài 3: Weighted Voting (có trọng số)

```
1 def knn_weighted(X_train, y_train, x_test, k=3):
2     distances = []
3     # Your code here #
```

# Dự Án Y Tế: Dự Đoán Bệnh Dựa Trên Triệu Chứng Sử Dụng KNN

*Hoàng-Nguyễn Vũ*

## 1. Mô tả dự án

**Tên dự án:** Dự đoán bệnh lý đơn giản từ triệu chứng sử dụng thuật toán K-Nearest Neighbors (KNN)

### 1.1 Mục tiêu

Mục tiêu của dự án là xây dựng một hệ thống phân loại bệnh đơn giản dựa trên các triệu chứng lâm sàng cơ bản của bệnh nhân. Hệ thống sử dụng thuật toán học máy KNN để dự đoán bệnh lý như cảm lạnh, cúm, hoặc COVID-19 từ một số đặc trưng đầu vào như: sốt, ho, mệt mỏi, và khó thở.

### 1.2 Bối cảnh ứng dụng

Trong môi trường y tế công đồng, đặc biệt ở vùng sâu vùng xa, việc sàng lọc ban đầu các triệu chứng phổ biến có thể giúp đưa ra hướng xử lý nhanh chóng và hỗ trợ quyết định lâm sàng ban đầu. Hệ thống KNN có thể đóng vai trò như một trợ lý thông minh để hỗ trợ y sĩ, sinh viên y khoa hoặc hệ thống chatbot tư vấn sức khỏe.

### 1.3 Lý do chọn KNN

KNN là một thuật toán đơn giản, dễ triển khai, không yêu cầu huấn luyện mô hình phức tạp. Nó phù hợp với bài toán nhỏ, có dữ liệu ít và mang tính chất phân loại. Trong trường hợp này, KNN sẽ giúp dự đoán loại bệnh dựa trên các triệu chứng có/không (nhị phân), với độ chính xác chấp nhận được trong phạm vi mô phỏng.

### 1.4 Cách tiếp cận

- Xây dựng một tập dữ liệu giả lập gồm các triệu chứng thường gặp và nhãn bệnh tương ứng.
- Mỗi bản ghi là một vector gồm 4 đặc trưng: Fever, Cough, Fatigue, Shortness of Breath.
- Áp dụng KNN để phân loại bệnh, với số lượng hàng xóm  $k$  có thể điều chỉnh.
- Dự đoán bệnh mới bằng cách so sánh với các bản ghi huấn luyện.

## 1.5 Kỳ vọng đầu ra

- Viết được hàm KNN cơ bản không dùng thư viện ngoài.
- Dự đoán đúng loại bệnh đối với bệnh nhân mới.
- Hiểu được cách KNN hoạt động trong thực tế và những giới hạn của nó.

## 2. Dữ liệu giả lập

Dữ liệu gồm 4 đặc trưng nhị phân và một nhãn phân loại:

Fever	Cough	Fatigue	Shortness of Breath	Disease
1	1	0	0	Cold
1	1	1	0	Flu
1	1	1	1	COVID-19
0	0	0	0	Healthy
0	1	0	0	Cold
1	0	1	1	COVID-19

## 3. Bài tập lập trình: Cài đặt thuật toán KNN

Yêu cầu:

1. Viết một hàm `euclidean_distance(x1, x2)` để tính khoảng cách Euclid giữa hai vector đầu vào.
2. Viết một hàm `knn_predict(X_train, y_train, x_test, k)` thực hiện:
  - Tính khoảng cách giữa `x_test` và mỗi phần tử trong `X_train`
  - Sắp xếp theo khoảng cách
  - Lấy `k` hàng xóm gần nhất
  - Trả về nhãn xuất hiện nhiều nhất trong các hàng xóm
3. Sử dụng dữ liệu sau để kiểm thử:

```

1 X_train = [
2   [1, 1, 0, 0], # Cold
3   [1, 1, 1, 0], # Flu
4   [1, 1, 1, 1], # COVID-19
5   [0, 0, 0, 0], # Healthy
6   [0, 1, 0, 0], # Cold
7   [1, 0, 1, 1], # COVID-19
8 ]
9
10 y_train = ["Cold", "Flu", "COVID-19", "Healthy", "Cold", "COVID-19"]
11

```

```
12 # Dự đoán triệu chứng:  
13 x_test = [1, 1, 1, 0] # Bệnh nhân mới  
14  
15 # Gợi ý:  
16 # print(knn_predict(X_train, y_train, x_test, k=3))
```

Kết quả mong đợi: Nhân dự đoán là "Flu" với  $k = 3$ .

# Dự Án Y Tế: Dự Đoán Chỉ Số Bệnh Tiểu Đường bằng KNN Hồi Quy

Hoàng-Nguyễn Vũ

## 1. Mô tả dự án

### 1.1 Bối cảnh

Tiểu đường là một căn bệnh mãn tính ngày càng phổ biến, ảnh hưởng đến hàng triệu người trên toàn thế giới. Việc theo dõi tiến triển của bệnh tiểu đường thông qua các chỉ số sinh học là yếu tố quan trọng giúp bác sĩ đưa ra quyết định điều trị sớm, chính xác và cá nhân hóa. Dự án này sử dụng một tập dữ liệu y khoa đã được chuẩn hóa từ **Viện nghiên cứu y khoa Hoa Kỳ** để mô phỏng quy trình dự đoán chỉ số tiến triển bệnh tiểu đường dựa trên các thông số lâm sàng của bệnh nhân.

### 1.2 Mục tiêu dự án

Xây dựng mô hình hồi quy bằng thuật toán **K-Nearest Neighbors (KNN Regression)** để dự đoán chỉ số tiến triển bệnh tiểu đường từ các đặc trưng đầu vào bao gồm:

- Tuổi
- Giới tính
- BMI (Chỉ số khối cơ thể)
- Huyết áp
- Các chỉ số sinh học khác (cholesterol, glucose, v.v.)

**Kết quả đầu ra** là một giá trị số liên tục biểu diễn mức độ nghiêm trọng của bệnh.

### 1.3 Bộ dữ liệu sử dụng

Dự án sử dụng bộ dữ liệu `load_diabetes()` từ thư viện `scikit-learn`, gồm:

- **442 bệnh nhân**, mỗi bệnh nhân có 10 chỉ số đầu vào (tất cả đã được chuẩn hóa).
- **Giá trị đầu ra (y)** là chỉ số định lượng đánh giá mức độ tiến triển của bệnh tiểu đường (diabetes progression).

### 1.4 Lý do chọn KNN Regression

Thuật toán KNN regression dễ cài đặt, không yêu cầu huấn luyện mô hình phức tạp và có thể cho kết quả cạnh tranh trong các bài toán y tế với tập dữ liệu vừa và nhỏ.

## 2. Bài tập lập trình

### Bài 1: Load và khám phá dữ liệu

```
1 from sklearn.datasets import load_diabetes
2 import matplotlib.pyplot as plt
3
4 data = load_diabetes()
5 X, y = data.data, data.target
6
7 # Your code here #
```

### Bài 2: Viết hàm KNN Regression (không dùng sklearn)

1. Viết hàm tính khoảng cách Euclidean
2. Viết hàm dự đoán: trung bình giá trị y của  $k$  hàng xóm gần nhất

### Bài 3: Tách tập huấn luyện và kiểm tra, đánh giá MAE/MSE

```
1 from sklearn.model_selection import train_test_split
2 from sklearn.metrics import mean_squared_error
3
4 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
5
6
7 # Your code here #
```

### Bài 4 (nâng cao):

- So sánh kết quả với KNN từ sklearn: KNeighborsRegressor
- Vẽ biểu đồ: MSE vs k
- Chuẩn hóa dữ liệu với StandardScaler

## 3. Kỳ vọng đầu ra

- Hiểu rõ cách áp dụng KNN Regression cho bài toán y tế.
- Cài đặt mô hình KNN từ đầu mà không phụ thuộc thư viện ngoài.
- Phân tích tác động của tham số  $k$  đến sai số dự đoán.