Blog Tuần 1 – Module 2

Khám phá NumPy, Đại số tuyến tính, MongoDB và Tư duy AI

GRID034

Ngày 14 tháng 7 năm 2025

Tóm tắt nội dung

Đây là chuỗi blog học thuật thuộc **Module 2: Ứng dụng NumPy, AI và Cơ sở dữ liệu NoSQL**.

Nội dung chính trong tuần bao gồm:

- **NumPy nâng cao:** Broadcasting, Indexing, Vectorization và bài toán phân loại học sinh.
- **Đại số tuyến tính ứng dụng:** Vector, ma trận, tích vô hướng, cosine similarity, ứng dụng xử lý ảnh.
- **MongoDB và NoSQL:** Cấu trúc document, truy vấn nâng cao, thao tác CRUD, kết nối với PyMongo.
- **Tư duy logic và giải quyết vấn đề:** 7 bước phân tích, khung MECE, Logic Tree và phân loại giải pháp.

Bài viết được thiết kế trực quan, đan xen giữa lý thuyết và thực hành Python, với các ví dụ cụ thể trong AI, giúp người học dễ tiếp cận và ứng dụng ngay vào dự án thật.

NumPy Căn Bản

Vũ Thái Sơn

Tóm tắt các kỹ thuật NumPy thường sử dụng và ứng dụng AI đơn giản

1. Giới thiệu về NumPy nâng cao và AI

NumPy là thư viện cốt lõi trong Python, không chỉ giúp xử lý mảng hiệu quả mà còn là nền tảng cho các ứng dụng trí tuệ nhân tạo (AI). Nếu bạn từng sắp xếp danh sách mua sắm hoặc tính tổng hóa đơn, bạn đã sử dụng những ý tưởng tương tự như các phép toán trên mảng mà NumPy cung cấp, nhưng nhanh hơn và mạnh mẽ hơn! Trong bài viết này, chúng ta sẽ khám phá các kỹ thuật NumPy nâng cao như *broadcasting*, *indexing*, và *vectorization*, sau đó áp dụng chúng vào một bài toán AI đơn giản: **phân loại điểm thi học sinh**. Mục tiêu là giúp bạn, dù mới học Python hay đã quen với AI, hiểu rõ cách NumPy biến những ý tưởng đơn giản thành công cụ mạnh mẽ, nên mình sẽ diễn đạt các kỹ thuật này ở mức đơn giản nhất.

2. Kỹ thuật NumPy nâng cao

2.1 Broadcasting: Phép thuật với mảng

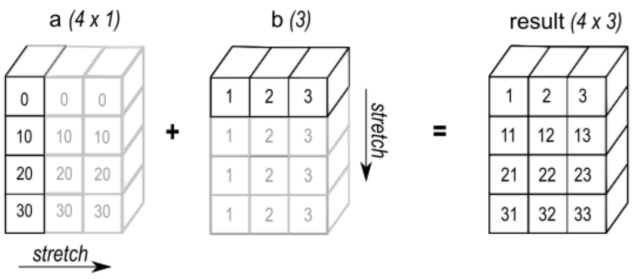
Hãy tưởng tượng bạn cần tăng giá tất cả món hàng trong siêu thị lên 10%. Thay vì cộng từng món, NumPy cho phép bạn thực hiện phép toán trên toàn bộ mảng cùng lúc nhờ *broadcasting*. Đây là kỹ thuật tự động mở rộng kích thước mảng để thực hiện phép toán mà không cần sao chép dữ liệu [1].

```
import numpy as np
prices = np.array([100, 200, 300])  # Prices of items
new_prices = prices * 1.1           # Increase by 10%
print(new_prices)                   # Output: [110. 220. 330.]
```

Listing 1: Ví dụ về broadcasting

Công thức: Nếu hai mảng có kích thước khác nhau, NumPy sẽ mở rộng mảng nhỏ hơn để khớp với mảng lớn hơn theo quy tắc:

Kích thước khớp nếu: (m, n) và $(1, n)$ hoặc $(m, 1)$



Hình 1: Minh họa broadcasting: Số 1.1 được nhân với từng phần tử của mảng.

2.2 Indexing thông minh: Lấy dữ liệu như cao thủ

Nếu bạn cần tìm tất cả học sinh có điểm trên 8 trong danh sách, NumPy cung cấp *indexing* để lọc dữ liệu nhanh chóng [2]. Có hai loại chính:

- **Boolean indexing**: Lọc dựa trên điều kiện.
- **Fancy indexing**: Sử dụng mảng chỉ số để truy cập.

```
scores = np.array([7, 8.5, 9, 6.5, 8])
high_scores = scores[scores > 8]  # Boolean indexing with condition > 8
print(high_scores)                # Output: [8.5, 9.0]
```

Listing 2: Boolean indexing lọc điểm cao

```
indices = np.array([0, 2, 4])      # Select specific indices
selected_scores = scores[indices]  # Fancy indexing with index array
print(selected_scores)             # Output: [7.  9.  8.]
```

Listing 3: Fancy indexing truy cập chỉ số

Phương pháp	Mô tả
Boolean indexing	Lọc dữ liệu theo điều kiện logic (>, <, ==).
Fancy indexing	Truy cập phần tử bằng danh sách chỉ số.

Bảng 1: So sánh các phương pháp indexing.

2.3 Vectorization: Tăng tốc mà không cần vòng lặp

Thay vì dùng vòng lặp để tính tổng bình phương điểm số, NumPy cho phép bạn *vector hóa* phép toán, đặc biệt hiệu quả với dữ liệu lớn [3]. Kỹ thuật này tăng tốc độ nhờ xử lý song song trên toàn bộ mảng cùng lúc, tận dụng tối đa sức mạnh của CPU/GPU, thay vì lặp qua từng phần tử như cách truyền thống. Với dữ liệu lớn, vectorization giảm đáng kể thời gian xử lý so với vòng lặp, nhờ loại bỏ chi phí gọi hàm và quản lý bộ nhớ tuần tự.

```
# Loop method
scores = np.array([7, 8, 9])
sum_squares = 0
for s in scores:
    sum_squares += s**2
print(sum_squares)  # Output: 194

# Vectorized method
sum_squares = np.sum(scores**2)
print(sum_squares)  # Output: 194
```

Listing 4: Vectorization so với vòng lặp

Lợi ích: Vectorization đặc biệt tối ưu với dữ liệu lớn nhờ xử lý song song, giúp giảm thời gian tính toán và tăng hiệu suất so với vòng lặp.

3. Ứng dụng AI: Phân loại điểm thi

Hãy áp dụng NumPy vào một bài toán AI đơn giản: phân loại học sinh dựa trên điểm thi. Giả sử bạn có điểm Toán, Lý, Hóa của 5 học sinh và cần dự đoán xem họ có đậu (1) hay trượt (0) dựa trên ngưỡng điểm trung bình.

3.1 Dữ liệu và bài toán

Dữ liệu gồm mảng 2D (5 học sinh, 3 môn) và nhãn (đậu/trượt). Chúng ta sẽ sử dụng *broadcasting* để thêm điểm thưởng, *indexing* để lọc dữ liệu, và *vectorization* để tính toán hiệu quả.

```
import numpy as np

# Data: Scores of 5 students in 3 subjects (Math, Physics, Chemistry)
data = np.array([[8, 7, 6], [9, 8, 9], [6, 5, 7], [7, 8, 7], [9, 9, 8]])

# Step 1: Apply broadcasting to add bonus points
bonus = np.array([0.5, 0.5, 0.5])  # Bonus points for each subject
data_with_bonus = data + bonus     # Broadcasting adds bonus to each
    element
print("Data with bonus:", data_with_bonus)  # Output: [[8.5 7.5 6.5] [9.5
    8.5 9.5] [6.5 5.5 7.5] [7.5 8.5 7.5] [9.5 9.5 8.5]]

# Step 2: Use indexing to calculate mean and filter high performers
mean_scores = np.mean(data_with_bonus, axis=1)  # Average for each student
high_performers = mean_scores[mean_scores > 7]  # Boolean indexing for high
     scores
print("High performers' average scores:", high_performers)  # Output:
    [7.83333333 9.16666667 7.83333333 9.16666667]

# Step 3: Use vectorization for classification and accuracy
labels = np.array([1, 1, 0, 1, 1])  # 1: Pass, 0: Fail
predictions = mean_scores > 7       # Vectorized threshold comparison
accuracy = np.mean(predictions == labels)  # Vectorized accuracy
    calculation
print(fĐộ" chính xác: {accuracy*100:.2f}%")  # Output: 80.00%
```

Listing 5: Mã nguồn hoàn chỉnh phân loại điểm thi

4. Kiến thức mở rộng

NumPy không chỉ dừng ở các kỹ thuật trên. Dưới đây là một số hướng phát triển:

- **Tích hợp với AI**: NumPy là nền tảng cho các thư viện như TensorFlow, PyTorch. Dữ liệu trong AI thường là mảng nhiều chiều, và NumPy xử lý chúng hiệu quả [4].
- **Tối ưu hóa**: Sử dụng np.einsum để tính toán tensor phức tạp hoặc np.linalg cho đại số tuyến tính.
- **Xử lý dữ liệu lớn**: Kết hợp NumPy với Dask hoặc CuPy để xử lý dữ liệu trên GPU.

Ví dụ thực tế: Trong xử lý ảnh AI, mỗi ảnh là một mảng 3D (chiều cao, chiều rộng, kênh màu). NumPy giúp tiền xử lý như chuẩn hóa giá trị pixel trước khi đưa vào mô hình học sâu.

5. Kết luận

NumPy là công cụ mạnh mẽ, giúp bạn từ những phép toán đơn giản như tính hóa đơn đến các bài toán AI phức tạp như phân loại. Với *broadcasting*, *indexing*, và *vectorization*, bạn có thể viết code nhanh, gọn, và hiệu quả. Bài toán phân loại điểm thi chỉ là bước đầu – hãy thử áp dụng NumPy vào dữ liệu thực tế như dự đoán thời tiết hoặc nhận diện ảnh! Tham khảo thêm tài liệu [5, 6] để củng cố kiến thức.

Tài liệu

[1] NumPy Developers. (2023) Broadcasting – numpy user guide. [Online]. Available: https://numpy.org/doc/stable/user/basics.broadcasting.html

[2] ——. (2023) Indexing on ndarrays – numpy user guide. [Online]. Available: https://numpy.org/doc/stable/user/basics.indexing.html

[3] B. Solomon. (2019) Look ma, no for loops: Array programming with numpy. Real Python tutorial. [Online]. Available: https://realpython.com/numpy-array-programming/

[4] Q.-D. Nguyen and V. D. Nguyen, *NumPy Arrays (Slides)*, AI Vietnam – AIO2025, 2025, bài giảng NumPy nâng cao (tiếng Việt).

[5] D.-T. Nguyen and Q.-V. Dinh, *NumPy Cơ Bản*, AI Vietnam – AIO2025, 2025, tài liệu nội bộ (tiếng Việt).

[6] Q.-V. Dinh, *Học List và NumPy Qua Các Ví Dụ*, AI Vietnam – AIO2025, 2025, tài liệu nội bộ (tiếng Việt, 29/06/2025).

Tìm Hiểu Numpy và Biểu Diễn Dữ Liệu 2D/3D Qua Ứng Dụng AI

Đàm Nguyên Khánh

Mục tiêu bài viết

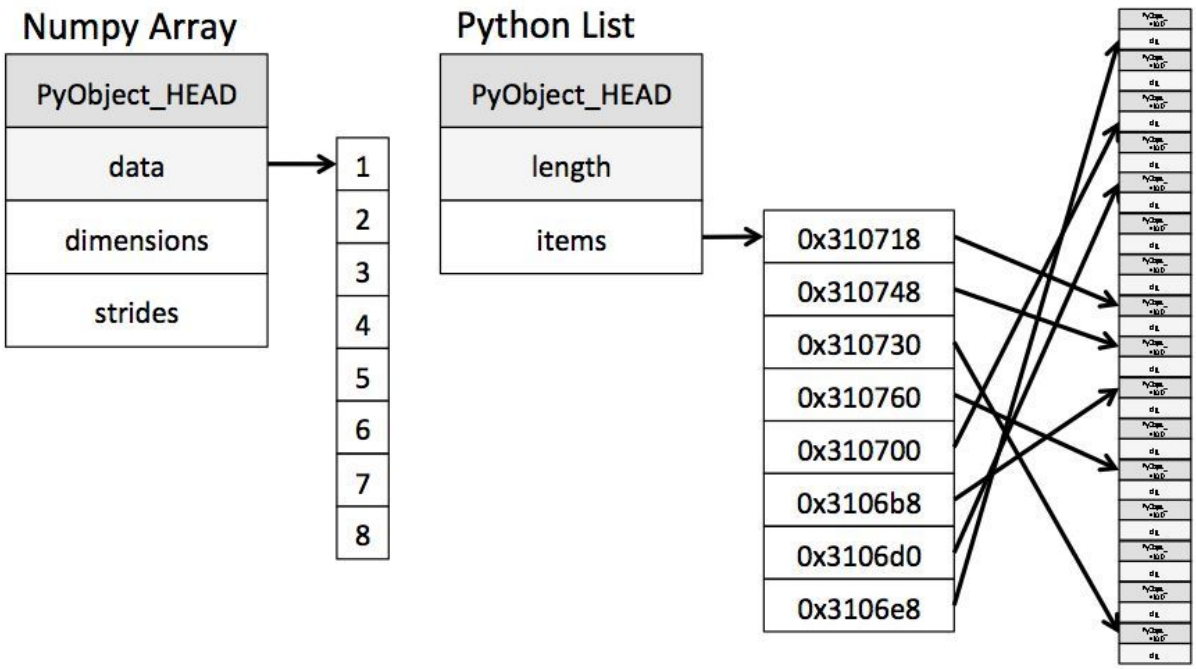
Trong bài viết này, bạn sẽ học cách sử dụng thư viện **NumPy** để làm việc với dữ liệu dạng mảng nhiều chiều, các thao tác phổ biến như *reshape*, *slicing*, *broadcasting*, và các ứng dụng thực tế trong **AI và xử lý ảnh**.

1. Giới thiệu về NumPy

- NumPy là thư viện Python chuyên dụng cho tính toán khoa học.
- Hỗ trợ mảng nhiều chiều ndarray.
- Tương thích tốt với OpenCV, Pandas, SciPy, Matplotlib.

Ví dụ tạo mảng:

```
import numpy as np
arr = np.array([1, 2, 3])
```



Hình 2: So sánh array trong Python list và NumPy array.

Mảng NumPy và danh sách Python thoạt nhìn có vẻ tương tự, nhưng chúng thuộc hai lớp đối tượng hoàn toàn khác nhau:

- **Mảng NumPy** (class ndarray):
  - Dữ liệu được lưu trữ trong **một khối bộ nhớ liền kề** (contiguous memory block).
  - Tính chất này được gọi là **tính cục bộ tham chiếu** (locality of reference), giúp truy cập và xử lý dữ liệu nhanh hơn.
  - Phù hợp cho việc xử lý **tập dữ liệu lớn** nhờ cấu trúc tối ưu.
- **Danh sách Python** (class list):
  - Dữ liệu **không được lưu trữ liền kề** trong bộ nhớ, mà nằm rải rác ở nhiều vị trí.
  - Do đó, việc truy cập và xử lý dữ liệu chậm hơn so với mảng NumPy.

2. Một số hàm thường dùng

Tạo mảng

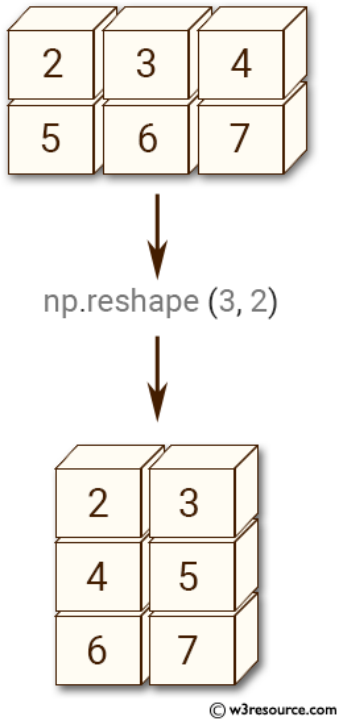
```
np.zeros((2, 3))
np.ones((2, 3))
np.arange(0, 10, 2)
```

Thay đổi cấu trúc

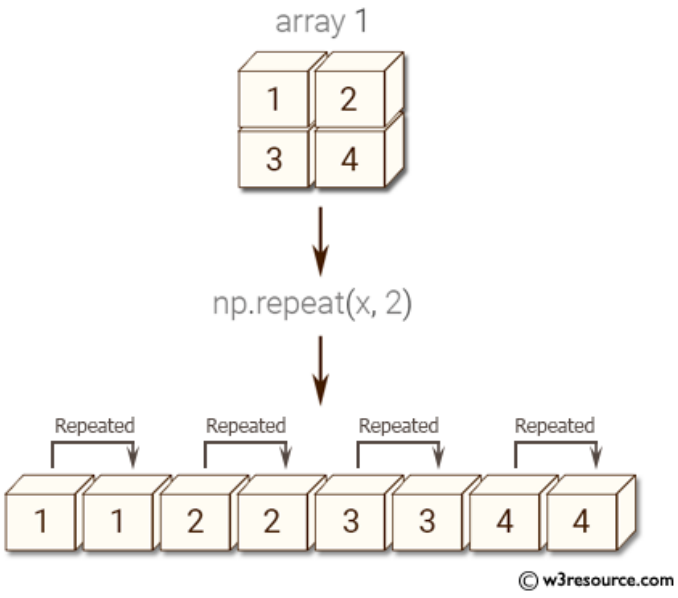
```
arr.reshape((3, 2))
arr.flatten()
```

Repeat theo chiều

```
np.repeat(data, 2, axis=0)
np.repeat(data, 2, axis=1)
```



(a) Hàm reshape

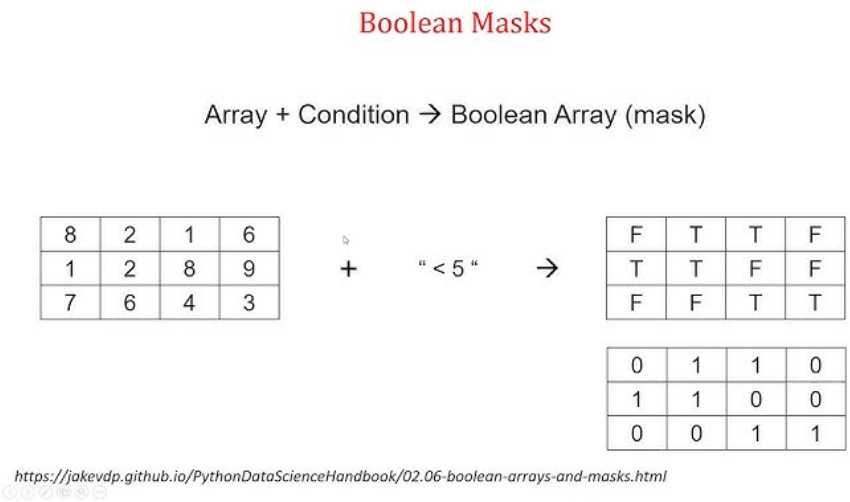


(b) Hàm repeat

3. Truy cập dữ liệu (Indexing)

- **Slicing:** arr[1:3, 0:2]
- **Row/column:** arr[0, :] vs arr[:, 1]
- **Boolean:** arr[arr > 3]

```
arr = np.array([[10, 20, 30],
                [40, 50, 60]])

col = arr[:, 1]
row = arr[1, :]
```

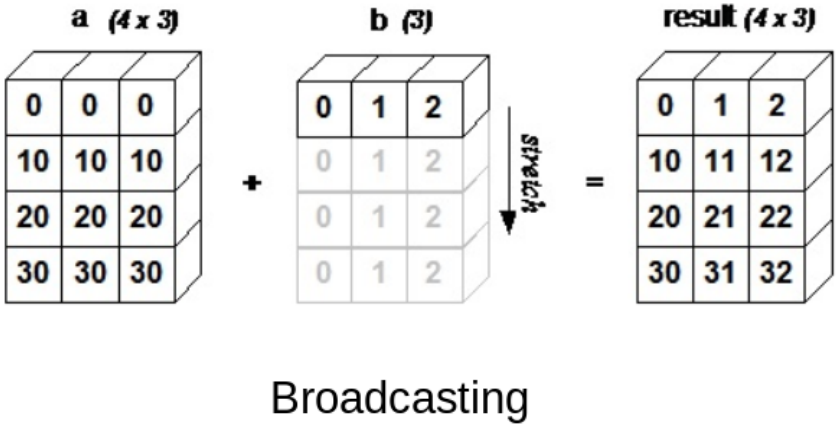


Hình 4: Chọn hàng, cột bằng slicing và boolean mask.

4. Broadcasting

NumPy hỗ trợ thực hiện phép toán giữa mảng và số vô hướng, hoặc giữa các mảng khác kích thước (nếu phù hợp).

```
data = np.array([[1, 2], [3, 4]])
result = data * 2
```

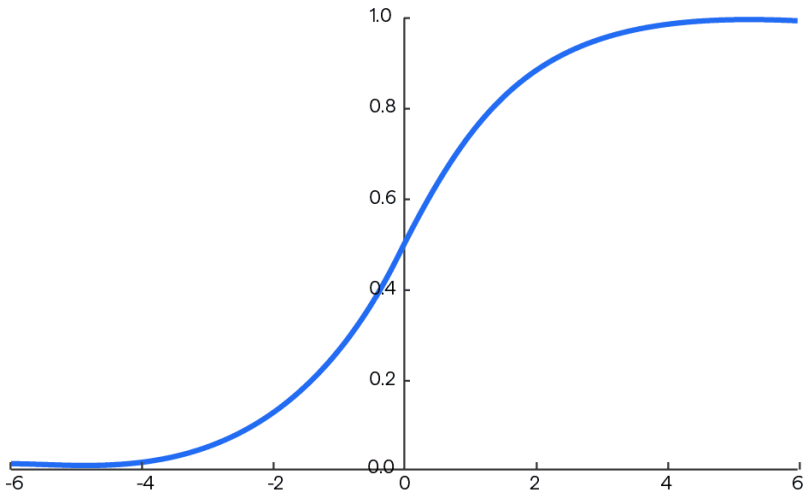


Hình 5: Broadcasting giữa ma trận và vector/scalar.

5. Hàm Softmax trong AI

Công thức ổn định:

$$f(x_i) = \frac{e^{x_i - \max(x)}}{\sum_j e^{x_j - \max(x)}}$$



Hình 6: Phân phối xác suất softmax

Ví dụ:

```
x = np.array([1.0, 2.0, 3.0])
e_x = np.exp(x - np.max(x))
softmax = e_x / e_x.sum()
```

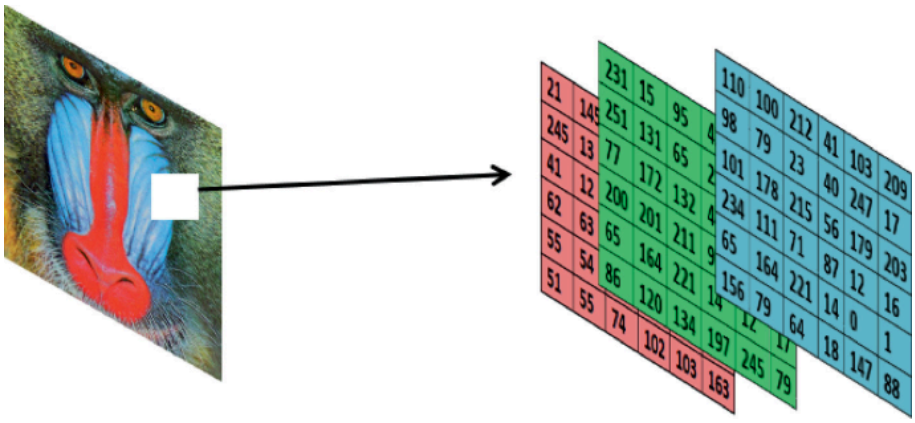
6. Xử lý ảnh với OpenCV và NumPy

Ảnh Grayscale và RGB

- Ảnh grayscale: mỗi pixel là số từ 0–255.
- Ảnh RGB: mỗi pixel là bộ 3 giá trị R, G, B.

Đọc ảnh với OpenCV

```
import cv2
img = cv2.imread("image.jpg")
img_rgb = img[:, :, ::-1]  # Chuyển từ BGR sang RGB
```

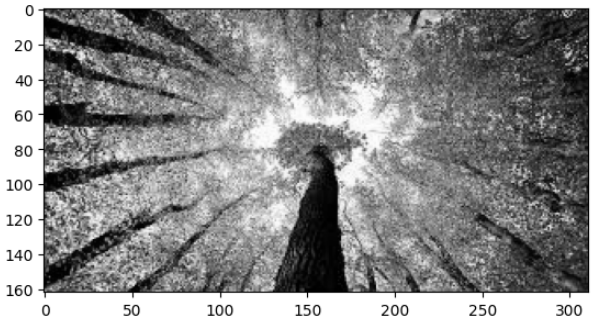


Hình 7: Pixel RGB và hiển thị ảnh

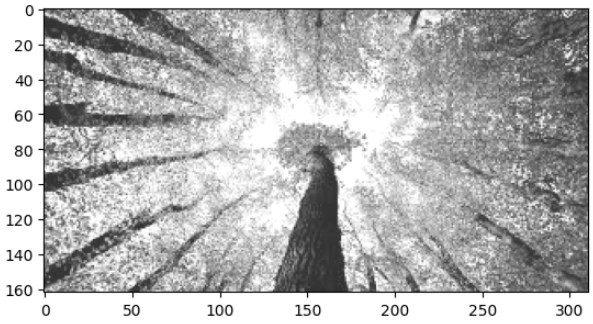
7. Điều chỉnh độ sáng ảnh

Tăng hoặc giảm độ sáng

```
img = img.astype(float)
img += 50       # hoặc img -= 50
img = np.clip(img, 0, 255)
img = img.astype(np.uint8)
```



(a) Ảnh gốc



(b) Ảnh tăng sáng

8. Một số ứng dụng thực tế

One-hot Encoding

```
np.eye(3)[2]  # Output: [0, 0, 1]
```

Xử lý dữ liệu thời tiết

```
temps = np.array([...])
reshaped = temps.reshape(6, 6)
daily_avg = np.mean(reshaped, axis=1)
```

Xử lý dữ liệu văn bản

```
labels = np.genfromtxt("data.csv", delimiter=",", usecols=4, dtype=str)
np.unique(labels)
```

Tổng kết

- **Tạo mảng:** zeros, ones, arange
- **Thay đổi cấu trúc:** reshape, flatten, repeat
- **Truy cập dữ liệu:** slicing, boolean indexing
- **Broadcasting:** thao tác với mảng khác kích thước
- **Ứng dụng AI:** xử lý ảnh, one-hot, softmax, thời tiết

Linear Algebra and Applications

Dao Lam Hoang

1. Vector và Matrix

1.1 Khái niệm

	Vector	**Matrix**
Khái niệm	Vector là một dãy có thứ tự các số, dùng để biểu diễn tọa độ trong không gian nhiều chiều.	Matrix là một bảng chữ nhật các số (sắp xếp thành hàng và cột), và là một cấu trúc mở rộng của vector.
Kí hiệu	$\vec{v} \in \mathbb{R}^n$	$A \in \mathbb{R}^{m \times n}$
Biểu diễn	$\vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \in \mathbb{R}^3$	$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \in \mathbb{R}^{m \times n}$

1.2 Các phép toán vector

1.2.1 Các phép toán

1 Phép toán cơ bản

Vector

- Phép cộng:

$$\vec{v} + \vec{u} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} + \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} v_1 + u_1 \\ v_2 + u_2 \\ \vdots \\ v_n + u_n \end{bmatrix}$$

- Phép trừ:

$$\vec{v} - \vec{u} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} - \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} v_1 - u_1 \\ v_2 - u_2 \\ \vdots \\ v_n - u_n \end{bmatrix}$$

```python
import numpy as np
X = np.array([5,6,7,8])
Y = np.array([1,2,3,4])
#ổTng
print(X+Y) # 6 8 10 12
print(np.add(X, Y)) # 6 8 10 12
# ệHiu
print(X-Y) # 4 4 4 4
print(np.substract(X, Y)) # 4 4 4 4
```

2 Phép toán khác

Vector

- Nhân vector với một số

$$c\vec{v} = c\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} cv_1 \\ cv_2 \\ \vdots \\ cv_n \end{bmatrix}$$

- Độ dài vector

$$\|\vec{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$$

```python
import numpy as np
data = np.array([1,2,4,2])

#Nhan so
data_2 = data*2 #[2, 4, 8, 4]

#chieu dai vector
data_length = np.linalg.norm(data) #5.0
```

3 Element-wise

Vector

- Tích Hadamard

$$\vec{v} \odot \vec{u} = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \odot \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} v_1 \times u_1 \\ \vdots \\ v_n \times u_n \end{bmatrix}$$

- Chia Hadamard

$$\vec{v} \oslash \vec{u} = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \oslash \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} \frac{v_1}{u_1} \\ \vdots \\ \frac{v_n}{u_n} \end{bmatrix}$$

```python
import numpy as np
x = np.array([1,2,3,4])
y = np.array([5,6,7,8])

#Element wise prod
prod = x*y

#Element wise div
div = x/y
```

4 Tích vô hướng

Vector

- Tích vô hướng

$$\vec{v} \cdot \vec{u} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = v_1 u_1 + v_2 u_2 + \cdots + v_n u_n$$

```python
import numpy as np
v = np.array([1,2])
w = np.array([2,3])
product = np.dot(v,w) #8
```

5 Min-max

Vector

- **Giá trị nhỏ nhất (min):**

$$\min(\vec{v}) = \min \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \text{giá trị nhỏ nhất trong } \vec{v}$$

- **Giá trị lớn nhất (max):**

$$\max(\vec{v}) = \max \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \text{giá trị lớn nhất trong } \vec{v}$$

- **Vị trí của giá trị nhỏ nhất (argmin):**

$$\text{argmin}(\vec{v}) = \text{chỉ số } i \text{ sao cho } v_i = \min(\vec{v})$$

- **Vị trí của giá trị lớn nhất (argmax):**

$$\text{argmax}(\vec{v}) = \text{chỉ số } i \text{ sao cho } v_i = \max(\vec{v})$$

```python
import numpy as np

# Vector v
v = np.array([10, 5, 7, 3, 9])

# Giá trị nhỏ nhất
print("Min:", np.min(v))          # Min: 3

# Giá trị lớn nhất
print("Max:", np.max(v))          # Max: 10

# Vị trí giá trị nhỏ nhất
print("Argmin:", np.argmin(v))    # Argmin: 3

# Vị trí giá trị lớn nhất
print("Argmax:", np.argmax(v))    # Argmax: 0
```

1.3 Các phép toán Matrix

1.3.1 Các phép toán

1 Phép toán cơ bản

Matrix

- Phép cộng

$$A + B = \begin{bmatrix} (a_{11} + b_{11}) & \cdots & (a_{1n} + b_{1n}) \\ \vdots & \ddots & \vdots \\ (a_{m1} + b_{m1}) & \cdots & (a_{mn} + b_{mn}) \end{bmatrix}$$

- Phép trừ

$$A - B = \begin{bmatrix} (a_{11} - b_{11}) & \cdots & (a_{1n} - b_{1n}) \\ \vdots & \ddots & \vdots \\ (a_{m1} - b_{m1}) & \cdots & (a_{mn} - b_{mn}) \end{bmatrix}$$

```python
import numpy as np

# ởKhi ạto hai ma ậtrn A và B (cùng kích ướthc 2x2)
A = np.array([
    [1, 2],
    [3, 4]
])

B = np.array([
    [5, 6],
    [7, 8]
])

# Phép ộcng ma ậtrn
sum_matrix = A + B
print("A + B =")
print(sum_matrix)

# Phép ừtr ma ậtrn
diff_matrix = A - B
print("A - B =")
print(diff_matrix)

# Output:
# A + B =
# [[ 6  8]
#  [10 12]]
#
# A - B =
# [[-4 -4]
#  [-4 -4]]
```

2 Matrix x vector

Matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$A\vec{x} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix} \in \mathbb{R}^m$$

```python
import numpy as np

# Ma âtrn A có kích ướthc 3x2
A = np.array([
    [1, 2],
    [3, 4],
    [5, 6]
])

# Vector x có kích ướthc 2x1
x = np.array([7, 8])

# Tính tích A.x
result = A @ x
print(result) #[23 53 83]
```

3 Matrix x Matrix

Matrix

Giả sử:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \in \mathbb{R}^{2\times 3}, \quad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \in \mathbb{R}^{3\times 2}$$

Tích:

$$A \cdot B = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} \end{bmatrix} \in \mathbb{R}^{2\times 2}$$

Mỗi phần tử của kết quả là tích vô hướng giữa hàng của A và cột tương ứng của B.

```python
import numpy as np

# Matrix A: shape (2, 3)
A = np.array([
    [1, 2, 3],
    [4, 5, 6]
])

# Matrix B: shape (3, 2)
```

```python
B = np.array([
    [7, 8],
    [9, 10],
    [11, 12]
])

# Matrix multiplication
C = A @ B
# ặhoc dùng C = np.matmul(A,B)
print("A @ B =")
print(C)

# Output:
# A @ B =
# [[ 58  64]
#  [139 154]]
```

1.4 Cosine simularity

Cosine Similarity

Định nghĩa: Cosine similarity đo lường mức độ tương tự giữa hai vector bằng cách tính cosin của góc giữa chúng.

$$\cos(\theta) = \frac{\vec{v} \cdot \vec{u}}{\|\vec{v}\| \cdot \|\vec{u}\|} = \frac{\sum_{i=1}^{n} v_i u_i}{\sqrt{\sum_{i=1}^{n} v_i^2} \cdot \sqrt{\sum_{i=1}^{n} u_i^2}}$$

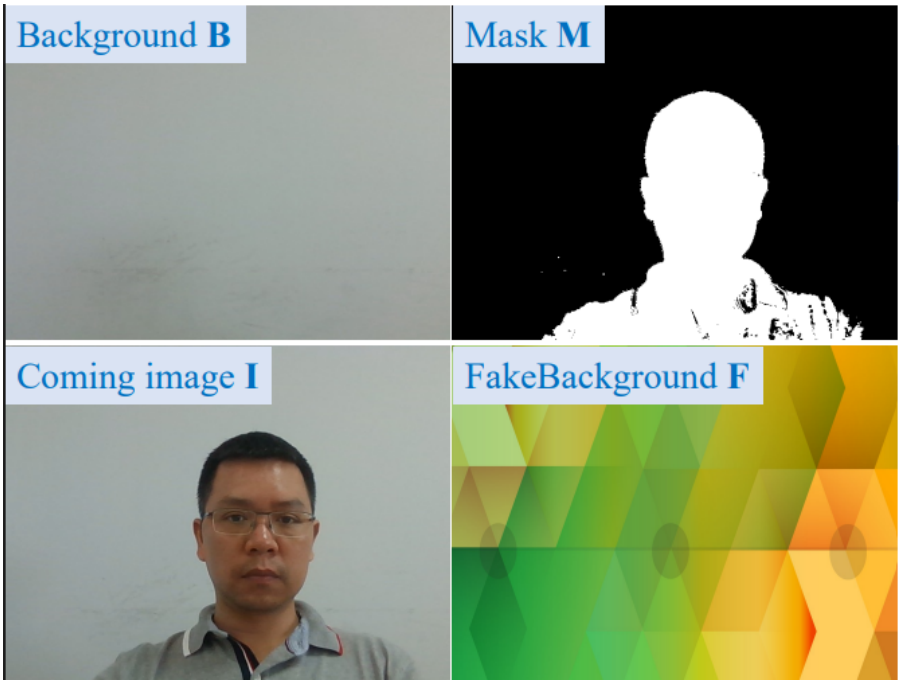
Tính chất:

- $-1 \leq \cos(\theta) \leq 1$
- cs($\vec{x}, \vec{y}$) = cs($a\vec{x}, b\vec{y}$), với ab > 0
- cs($\vec{x}, \vec{y}$) $\neq$ cs($\vec{x}$ + c, $\vec{y}$ + d)

```python
import numpy as np

# Hai vector
v = np.array([1, 2, 3])
u = np.array([4, 5, 6])

# Tính cosine similarity
cos_sim = np.dot(v, u) / (np.linalg.norm(v) * np.linalg.norm(u))

print("Cosine Similarity:", cos_sim)

# Output:
# Cosine Similarity: 0.974631846
```

2. Ứng dụng

2.1 Background Substraction



Hình 9: Sample

1. Ý tưởng

- So sánh sự khác nhau giữa hai ảnh B và I
- Ta sẽ tìm một mức điểm để phân biệt các pixel gần giống nhau gắn giá trị 0 và khác nhau để mask
- Ta ghép lại ảnh I vào F bằng cách thay các pixel của F bằng các pixel có giá trị 0 sau khi mask, còn lại là pixel I.

Background Substraction

$$A = |I - B|$$

$$M = \begin{cases} 0 & \text{if } A < th \\ 1 & \text{otherwise} \end{cases}$$

$$O = \begin{cases} F & \text{if } M == 0 \\ I & \text{otherwise} \end{cases}$$

2. Code

```
import numpy as np
import cv2

```

```
bg = cv2.imread('background.png', 1)
bg = cv2.resize(bg, (640, 480))

img = cv2.imread('StillImage.png', 1)
img = cv2.resize(img, (640, 480))

difference = cv2.absdiff(bg, img)
difference = np.sum(differnce, axis = 2)/3.0 #ấly trung bình ữgia 3 ớlp R,G, B

_, difference_binary = cv2.threshold(difference, 15, 255, cv2.THRESH_BINARY)

new_bg = cv2.imread('FakeBackground.png')
new_bg = cv2.resize(new_bg, (640, 480))

output = np.where(difference_binary==0, new_bg, img)
cv2.imwrite('output.png', output)
```

2.2 Traffic Sign Matching

Hình 10: Sample

Ý tưởng sử dụng L1-norm và hàm cosine để kiểm tra sự giống nhau giữa 2 ảnh.

1. L1-norm

```
import cv2
import matplotlip.pyplot as plt

#read img
img1 = cv2.imread('sign1.png')
img2 = cv2.imread('sign2.png')
```

```
#resize img
img1 = cv2.resize(img1,(100,100))
img2 = cv2.resize(img2,(100,100))s
# doi type de tinh toan
img1 = img1.astype(np.float32)
img 2 = img2.astype(np.float32)
distance = np.mean(np.abs(img1 - img2))
print(distance)
```

2. Cosine simularity

```
from numpy.linalg import norm

def cos_sim(x, y):
    x = x.flatten()
    y = y.flatten()

    x = x.astype(np.float64)
    y = y.astype(np.float64)

    result = np.dot(x, y) / (norm(x) * norm(y))
    return result

cs12 = cos_sim(img1, img2) #gia tri tra ra tu -1 den 1
```

Tư Duy Logic và Giải Quyết Vấn Đề trong Dự Án AI & Data Science

Đàm Nguyên Khánh

1. Giới thiệu

Trong thời đại AI bùng nổ, việc xây dựng hệ thống thông minh không chỉ dừng lại ở thuật toán. Điều quan trọng hơn là tư duy logic và kỹ năng giải quyết vấn đề một cách có hệ thống để đảm bảo **AI tạo ra giá trị thực sự** cho doanh nghiệp.

Tại sao cần học Logical Thinking & Problem Solving?

- Xác định đúng vấn đề là yếu tố then chốt quyết định 80% thành công dự án AI.
- Tối ưu hóa nguồn lực, tránh đầu tư sai hướng.
- Phân biệt giữa hype và giá trị thật của AI.

2. Phần I: Nền tảng tư duy giải quyết vấn đề trong AI

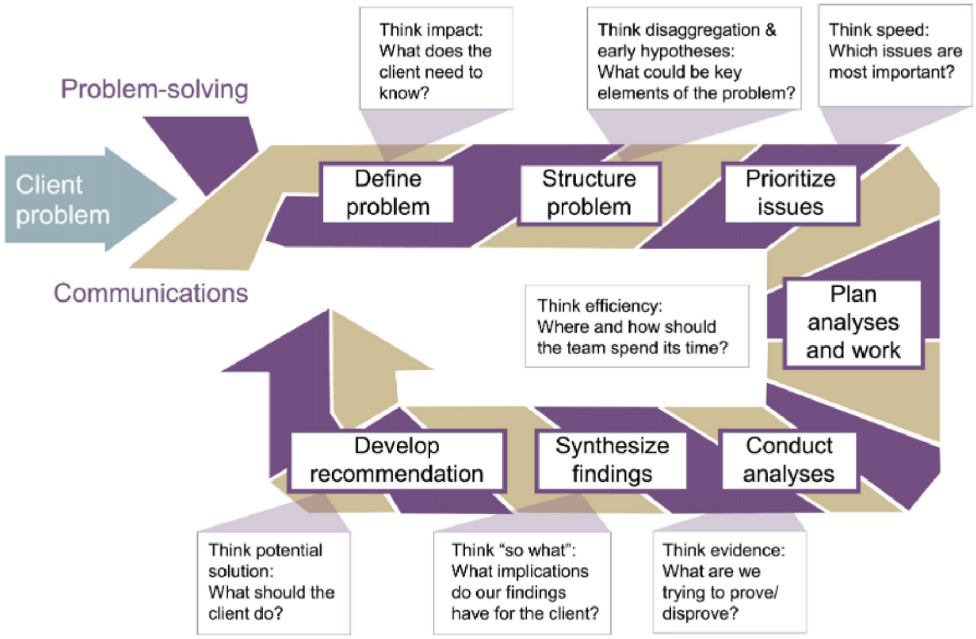
2.1 Framework 7 bước giải quyết vấn đề

1. Xác định vấn đề
2. Chia nhỏ vấn đề (MECE)
3. Ưu tiên vấn đề (Impact vs Feasibility)
4. Thu thập dữ liệu
5. Phân tích dữ liệu
6. Đề xuất giải pháp
7. Triển khai và đánh giá

PALOS
VENTURE ON

7 BƯỚC XỬ LÝ VẤN ĐỀ NHANH CHÓNG

THEO MÔ HÌNH PROBLEM-SOLVING CỦA MCKINSEY



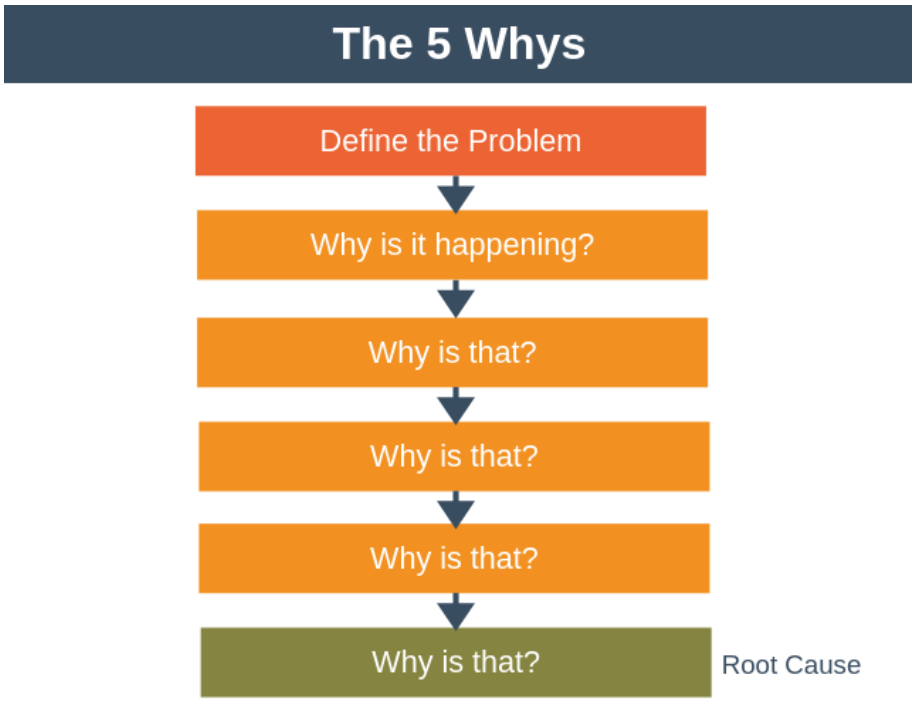
Hình 11: Sơ đồ 7 bước giải quyết vấn đề trong AI

2.2 Kỹ thuật xác định vấn đề

5W1H: Hỏi đúng giúp nhìn rõ toàn cảnh vấn đề:

- **What:** Vấn đề là gì? (VD: CTR thấp hơn benchmark)
- **Why:** Tại sao quan trọng? (Tác động đến doanh thu)
- **Who, Where, When, How**...

5 Whys: Đào sâu tới nguyên nhân gốc rễ. *Ví dụ: Doanh số giảm → sản phẩm hết hàng → quản lý kho không hiệu quả → thiếu dự báo → chưa đầu tư BI → chưa nhận thức tầm quan trọng.*



Hình 12: 5 Whys

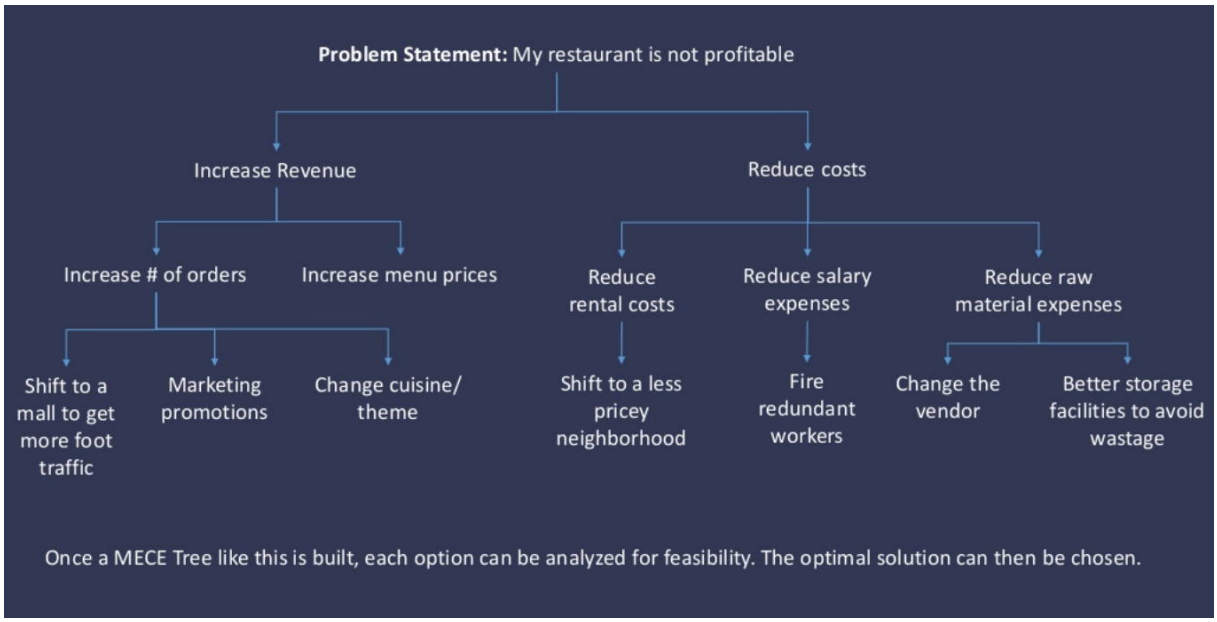
SMART Framework: Giúp định nghĩa mục tiêu đúng:

- Specific – Cụ thể
- Measurable – Đo lường được
- Achievable – Có thể đạt được
- Relevant – Liên quan đến mục tiêu kinh doanh
- Time-bound – Có hạn định rõ ràng

3. Phần II: Phân tích vấn đề và ưu tiên giải pháp

3.1 MECE và Logic Tree

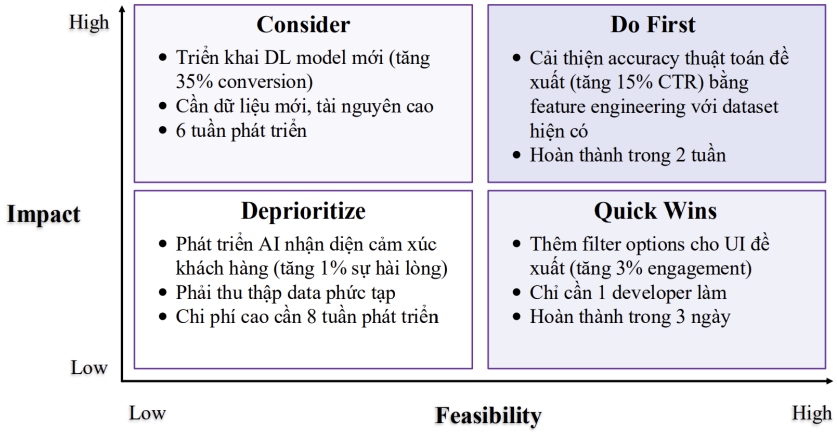
- **MECE (Mutually Exclusive, Collectively Exhaustive):** Tránh trùng lặp, bỏ sót.
- **Logic Tree:** Biểu diễn cấu trúc nguyên nhân-hệ quả rõ ràng.



Hình 13: Logic Tree áp dụng MECE để giải quyết vấn đề

3.2 Ưu tiên với ma trận Impact–Feasibility

- Đánh giá giải pháp theo Tác động (Impact) và Khả thi (Feasibility)
- Phân loại:
  - **Do First**: High Impact – High Feasibility
  - **Quick Wins**: Low Impact – High Feasibility
  - **Consider**: High Impact – Low Feasibility
  - **Deprioritize**: Low–Low



Hình 14: Logic Tree áp dụng MECE để giải quyết vấn đề

4. Phần III: Dữ liệu và giải pháp trong dự án AI

4.1 Thu thập dữ liệu hiệu quả

- Phân loại dữ liệu: định lượng, định tính, hành vi
- Nguồn: logs, phỏng vấn, A/B test, dữ liệu ngành
- Đánh giá chất lượng dữ liệu: chính xác, đầy đủ, nhất quán, hợp lệ, kịp thời, duy nhất

Bảng 2: Tiêu chí chất lượng dữ liệu

Tiêu chí	Mô tả
Tính chính xác	Dữ liệu phản ánh đúng thực tế; ví dụ: địa chỉ khách hàng khớp với hệ thống bưu chính.
Tính đầy đủ	Bao gồm toàn bộ thông tin cần thiết; ví dụ: không thiếu giá trị ở các trường quan trọng.
Tính nhất quán	Dữ liệu đồng nhất trên các hệ thống hoặc bảng dữ liệu khác nhau.
Tính kịp thời	Dữ liệu được cập nhật đúng thời điểm cần thiết; ví dụ: tồn kho thời gian thực thay vì cập nhật định kỳ.
Tính hợp lệ	Tuân thủ các quy tắc nghiệp vụ hoặc định dạng chuẩn; ví dụ: tuổi trong khoảng 0–120, email đúng định dạng.
Tính duy nhất	Không có bản ghi trùng lặp; ví dụ: mỗi khách hàng có một định danh duy nhất.

4.2 Phân tích và đề xuất giải pháp

– **Làm sạch dữ liệu (40% thời gian)**: Xử lý dữ liệu thô: loại bỏ ID trùng lặp, xử lý thời gian bắt buộc, chuẩn hóa timestamp, áp dụng quy tắc nghiệp vụ.

– **Phân tích khám phá dữ liệu (30% thời gian)**: Phân tích histogram hàng ví, trung bình giá/CTR, phân khúc user (RFM), xác định mẫu hành vi theo mùa.

– **Đánh tích chuẩn đoán (20% thời gian)**: Phân tích giá CTR, kiểm tra hiệu quả thời gian xem/mua, A/B testing, đánh giá tác động thời gian hiển thị.

– **Tạo ra hiệu biệt hành động (10% thời gian)**: Đề xuất tham số tối ưu, thiết lập roadmap cải tiến, truyền đạt kết quả, xây dựng dashboard theo dõi KPI.

5. Tổng kết

- 80% thất bại AI đến từ định nghĩa vấn đề sai
- Framework 7 bước giúp giải quyết bài toán AI một cách có hệ thống
- MECE & Logic Tree giúp phân tích rõ ràng – không bỏ sót
- Ma trận ưu tiên giúp tập trung nguồn lực hiệu quả

- Giải pháp AI không thể rời dữ liệu chất lượng và hiểu đúng nhu cầu kinh doanh

NoSQL và MongoDB

Bùi Đức Xuân

1. Giới Thiệu

Chào mừng các bạn đến với blog học tập của mình! Hôm nay, chúng ta sẽ cùng nhau khám phá một chủ đề cực kỳ thú vị và quan trọng trong ngành Công nghệ thông tin: **Cơ sở dữ liệu NoSQL** và cụ thể là **MongoDB**.

2. Cơ Sở Dữ Liệu Là Gì? NoSQL Khác Biệt Ra Sao?

- **Dữ liệu** là tập hợp thông tin: số, văn bản, hoặc phương tiện khác.
- **Cơ sở dữ liệu** là tập hợp có cấu trúc, có thể truy cập, khôi phục và cập nhật.
- **Hệ quản trị cơ sở dữ liệu (DBMS)**: phần mềm tổ chức, kiểm soát truy cập và thao tác trên dữ liệu.
- **SQL** lưu trữ theo dạng bảng (quan hệ). Ví dụ:

```
Student: studentID, studentName
Mark: studentID, subjectID, mark
```

- **NoSQL** (Not Only SQL): lưu trữ phi-bảng, thường là cặp "key":"value".

3. Các Loại Cơ Sở Dữ Liệu NoSQL Phổ Biến

1. **Key-Value Store**:
   - Bảng băm đơn giản: truy cập qua khóa chính.
   - Ví dụ: Key: "Name", Value: "Thai"
2. **Graph Database**:
   - Dữ liệu lưu dưới dạng *node*, *edge*, *attribute*.
   - Ví dụ: Thai --Is_Friend_With--> Anh
3. **Document Database**:
   - Dữ liệu dạng XML, JSON, BSON.
   - Mỗi document định danh bởi một khóa duy nhất.

4. MongoDB – Document Database Tiêu Biểu

Cấu trúc dữ liệu trong MongoDB

- Lưu trữ field-value trong **documents**.
- Các documents nằm trong **collections**, collections nằm trong **database**.

- Định dạng chính: **JSON**.
- Mỗi document có _id là định danh duy nhất.
- Hỗ trợ:
  - **Nested Documents**
  - **Arrays**
  - Cấu trúc linh hoạt (đa hình)

Hệ sinh thái MongoDB

- Phiên bản: Community, Enterprise, Atlas (cloud).
- Công cụ hỗ trợ: Shell, Drivers, Compass, BI Connectors.

5. Cài Đặt MongoDB

- **MongoDB Atlas**: tạo Free Cluster.
- **Mongo Shell & Tools**: thao tác dòng lệnh.
- **MongoDB Compass**: giao diện GUI.

6. Ngôn Ngữ Truy Vấn MongoDB (MQL)

1. Tạo/Xóa Database và Collection

```
show dbs
use database_name
db.dropDatabase()
show collections
db.createCollection('collection_name')
db.collection_name.drop()
```

2. Chèn Dữ Liệu

```
db.student.insert({"name":"Thai"})
db.student.insert([{"name":"Thai"}, {"age":20}])
```

3. Nhập/Xuất Dữ Liệu

```
mongoexport --uri="<uri>" --collection=<collection_name> --out=<file>
mongoimport --uri="<uri>" --collection=<collection_name> --file=<file>
```

4. Tìm Kiếm

```
db.student.findOne({"name":"Anh"})
db.student.find({"name":"Anh", "age":20})
```

5. Toán Tử So Sánh

- $eq, $ne, $lt, $lte, $gt, $gte, $in, $nin

```
db.student.find({"salary": {$lt: 50000}})
```

6. Toán Tử Logic

```
db.students.find({
  $and: [{ "age" : 20 }, { "mark" : { $gt: 8.0 } }]
})
```

7. $expr

```
db.routes.find({$expr: {$eq:["$src_airport","$dst_airport"]}})
```

8. Toán Tử $exists và $type

```
db.companies.find({"ipo":{$exists:true}})
db.companies.find({"homepage_url":{$type:2}})
```

9. Cursor Methods

```
.count(), .limit(X), .skip(X), .sort({field:1/-1}), .size()
```

10. Projection

```
db.grades.find({}, {"student_id":1,"class_id":1, "_id":0})
```

11. Truy Vấn Embedded Documents

```
db.inspections.find({"address.zip":11385})
```

12. Truy Vấn Mảng

```
$all, $size, $elemMatch

db.students.find({ skills: { $all: ["math", "english"] } })
db.students.find({ hobbies: { $size: 2 } })
db.grades.find({"scores": {$elemMatch:{"type":"exam","score":{$gt:80}}}})
```

13. Xóa Dữ Liệu

```
db.student.deleteOne({"name":"Anh"})
db.student.deleteMany({"name":"Anh"})
```

14. Cập Nhật Dữ Liệu

```
db.student.updateOne({"name":"Thai"}, {$set:{"name":"Anh"}})
```

15. Toán Tử Cập Nhật

```
$set, $unset, $rename, $inc, $push

db.student.updateMany({"name":"Hoa"}, {$set:{"address":"HCM"}})
db.student.updateMany({}, {$unset:{"address":""}})
db.student.updateMany({}, {$rename:{"hobbies_list":"hobbies"}})
db.student.updateMany({"age":20}, {$inc:{"age":1}})
db.student.updateMany({"name":"Hoa"}, {$push:{"address":"Ha Noi"}})
```

7. Tổng Kết

- **NoSQL** không chỉ là ”No Relational” mà còn là ”Not Only SQL”.
- Có nhiều loại: *Key-value, Graph, Document*.
- **MongoDB** là đại diện nổi bật của Document Database, với cấu trúc linh hoạt và dễ mở rộng.
- Hệ thống truy vấn MQL mạnh mẽ hỗ trợ đầy đủ thao tác CRUD và các toán tử nâng cao.