NumPy - Tổng hợp các phương thức thông dụng và nâng cao cho Khoa học Dữ liệu và Trí tuệ Nhân tạo

duckAI soạn cho Hùng

11:13 AM +07, Thứ Năm, 24/07/2025

Mục lục

1 Giới thiệu **3**
1.1 Mục tiêu học tập . . . 3
1.2 Yêu cầu kiến thức . . . 3

2 Các hàm cơ bản để khởi tạo mảng và sinh dữ liệu **3**
2.1 np.array . . . 3
2.2 np.zeros . . . 4
2.3 np.ones . . . 4
2.4 np.eye . . . 4
2.5 np.arange . . . 5
2.6 np.linspace . . . 5
2.7 np.random . . . 6

3 Thuộc tính của mảng **6**
3.1 shape . . . 6
3.2 ndim . . . 7
3.3 size . . . 7
3.4 dtype . . . 7

4 Đánh chỉ số và cắt lát **8**
4.1 Truy cập phần tử cụ thể . . . 8
4.2 Truy xuất hàng/cột . . . 8
4.3 Lọc điều kiện . . . 8
4.4 np.where . . . 9
4.5 np.isnan, np.isinf . . . 9
4.6 np.clip . . . 10

5 Các phép toán và thống kê **10**
5.1 np.sum . . . 10
5.2 np.mean, np.std . . . 11
5.3 np.min / np.max . . . 11
5.4 np.median, np.percentile . . . 12
5.5 np.corrcoef . . . 12
5.6 np.cov . . . 13

5.7 np.histogram 13

6 Thao tác hình dạng và Reshape **14**
6.1 reshape 14
6.2 flatten 14
6.3 transpose (T) 14
6.4 np.concatenate 15
6.5 np.vstack, np.hstack 15
6.6 np.unique 16
6.7 np.vectorize 16

7 Sắp xếp và Tìm kiếm **17**
7.1 np.sort 17
7.2 np.argsort 17
7.3 np.argmax / np.argmin 17

8 Broadcasting và Đại số tuyến tính **18**
8.1 Broadcasting 18
8.2 np.dot 18
8.3 np.linalg 19
8.4 np.einsum 19

9 Xử lý dữ liệu lớn **20**
9.1 np.save, np.load 20

10 Ví dụ thực hành từ Kaggle **20**
10.1 Bài 1: Chuẩn hóa dữ liệu nhiệt độ 21
10.2 Bài 2: Tìm ngày doanh thu cao nhất 21
10.3 Bài 3: Tính toán ma trận 22
10.4 Bài 4: Phân tích dữ liệu phân loại 22
10.5 Bài 5: Xử lý dữ liệu thiếu 22
10.6 Bài 6: Phân tích phân phối dữ liệu 23
10.7 Bài 7: Tính toán tensor 23

1 Giới thiệu

Tài liệu này được thiết kế như một tài liệu nghiên cứu chuyên sâu về thư viện NumPy, một công cụ cốt lõi trong việc xử lý dữ liệu số và phát triển các ứng dụng trong Khoa học Dữ liệu và Trí tuệ Nhân tạo (AI). NumPy cung cấp các cấu trúc dữ liệu hiệu quả như mảng đa chiều (ndarray) và các hàm toán học tối ưu hóa, giúp tăng tốc độ tính toán và hỗ trợ các thuật toán phức tạp. Tài liệu nhằm mục đích phục vụ các học viên, từ người mới bắt đầu đến các nhà nghiên cứu, với các ví dụ thực tiễn và hướng dẫn chi tiết, bao gồm cả các bài tập lấy cảm hứng từ nền tảng Kaggle. Mỗi phần đều được xây dựng với mục tiêu cung cấp nền tảng lý thuyết kết hợp thực hành, đảm bảo học viên có thể áp dụng kiến thức vào các dự án thực tế.

1.1 Mục tiêu học tập

- Hiểu và sử dụng các phương thức cơ bản để khởi tạo và thao tác mảng NumPy. - Thành thạo các kỹ thuật thống kê, đại số tuyến tính và xử lý dữ liệu nâng cao. - Áp dụng NumPy trong các kịch bản thực tế như phân tích dữ liệu, học máy và mô phỏng.

1.2 Yêu cầu kiến thức

- Đã học ngôn ngữ lập trình Python cơ bản.

2 Các hàm cơ bản để khởi tạo mảng và sinh dữ liệu

2.1 np.array

Chức năng: Phương thức này chuyển đổi danh sách, tuple hoặc các cấu trúc iterable thành mảng NumPy (ndarray), hỗ trợ tính toán vector hóa với hiệu suất cao, rất quan trọng trong các ứng dụng khoa học dữ liệu.
Cú pháp: `np.array(object, dtype=None, copy=True)`
Tham số:

- **object:** Dữ liệu đầu vào (danh sách, tuple, iterable).
- **dtype:** Kiểu dữ liệu (ví dụ: `int32`, `float64`, `bool`), được suy ra tự động nếu không chỉ định.
- **copy:** Giá trị Boolean, xác định liệu có sao chép dữ liệu (True) hay sử dụng tham chiếu (False).

Ví dụ: Tạo mảng 1D và 2D với kiểu `float64`.

```
>>> import numpy as np
# khoi tao mang 1D NumPy voi cac phan tu [1.0, 2.0, 3.0] su dung kieu float64
# ho tro tinh toan vector hoa hieu qua trong phan tich so hoc
>>> arr1 = np.array([1, 2, 3], dtype=float)
# tao mang 2D NumPy [[1.0, 2.0], [3.0, 4.0]] voi kieu float64
# phu hop cho cac tinh toan ma tran trong khoa hoc du lieu
>>> arr2 = np.array([[1, 2], [3, 4]], dtype=float)
>>> print(arr1, "\n", arr2)
```

2.2 np.zeros

Chức năng: Tạo mảng chứa toàn số 0, thường được sử dụng để khởi tạo mảng trước khi thực hiện các phép tính phức tạp, đặc biệt trong các thuật toán học máy như khởi tạo ma trận trọng số.
Cú pháp: `np.zeros(shape, dtype=float, order='C')`
Tham số:

- **shape:** Tuple chỉ kích thước (ví dụ: `(2,3)` cho ma trận 2x3).
- **dtype:** Kiểu dữ liệu (mặc định là `float64`).
- **order:** Quy định cách lưu trữ dữ liệu theo hàng (`'C'`) hoặc cột (`'F'`).

Ví dụ: Tạo ma trận 2x3 chứa số 0.

```
>>> zerosArr = np.zeros((2, 3), dtype=int)
# tao ma tran 2x3 day so 0 su dung kieu integer
# thuong dung de khoi tao mang trong thuat toan nhu ma tran trong so mang nho dan
>>> print(zerosArr)
# hien thi ma tran ket qua: [[0 0 0], [0 0 0]]
```

2.3 np.ones

Chức năng: Tạo mảng chứa toàn số 1, hữu ích trong việc khởi tạo mảng hoặc thực hiện các phép nhân ma trận trong các bài toán toán học và học máy.
Cú pháp: `np.ones(shape, dtype=float, order='C')`
Tham số: Tương tự như `np.zeros`.
Ví dụ: Tạo ma trận 3x2 chứa số 1 kiểu `int`.

```
>>> onesArr = np.ones((3, 2), dtype=int)
# tao ma tran 3x2 day so 1 su dung kieu integer
# thuong dung lam diem bat dau cho cac phep cong tich luy hoac thao tac quy mo
>>> print(onesArr)
# hien thi ma tran ket qua: [[1 1], [1 1], [1 1]]
```

2.4 np.eye

Chức năng: Tạo ma trận đơn vị với các phần tử trên đường chéo chính bằng 1 và các phần tử khác bằng 0, đóng vai trò quan trọng trong đại số tuyến tính và các phép biến đổi ma trận.
Cú pháp: `np.eye(N, M=None, k=0, dtype=float)`
Tham số:

- **N:** Số hàng.
- **M:** Số cột (mặc định bằng `N`).
- **k:** Vị trí đường chéo (0: chính, dương: trên, âm: dưới).

- **dtype:** Kiểu dữ liệu.

Ví dụ: Tạo ma trận đơn vị 3x3.

```
>>> eyeArr = np.eye(3, dtype=int)
# tao ma tran don vi 3x3 su dung kieu integer
# mot ma tran co ban trong dai so tuyen tinh dung cho cac bien doi danh thuc
>>> print(eyeArr)
# hien thi ma tran ket qua: [[1 0 0], [0 1 0], [0 0 1]]
```

2.5 np.arange

Chức năng: Tạo mảng 1D chứa dãy số cách đều trong khoảng `[start, stop)`, thường được sử dụng để tạo các chuỗi số cho mô phỏng hoặc phân tích dữ liệu.
Cú pháp: `np.arange(start, stop, step, dtype=None)`
Tham số:

- **start:** Giá trị bắt đầu (bao gồm, mặc định 0).
- **stop:** Giá trị kết thúc (không bao gồm).
- **step:** Khoảng cách giữa các phần tử (mặc định 1).
- **dtype:** Kiểu dữ liệu.

Ví dụ: Tạo dãy số chẵn từ 0 đến 10.

```
>>> rangeArr = np.arange(0, 11, 2)
# tao mang 1D voi cac so chan [0, 2, 4, 6, 8, 10] voi buoc nhay 2
# huu ich cho viec tao day so trong mo phong hoac phan tich thoi gian
>>> print(rangeArr)
# hien thi mang ket qua: [0 2 4 6 8 10]
```

2.6 np.linspace

Chức năng: Tạo mảng 1D chứa một số lượng cố định (`num`) các giá trị chia đều trong khoảng `[start, stop]`, phù hợp cho việc tạo các điểm lấy mẫu đều trong các mô hình toán học.
Cú pháp: `np.linspace(start, stop, num=50, endpoint=True, dtype=None)`
Tham số:

- **start:** Giá trị đầu.
- **stop:** Giá trị cuối (bao gồm nếu `endpoint=True`).
- **num:** Số phần tử.
- **endpoint:** Giá trị Boolean, xác định có bao gồm `stop` không.
- **dtype:** Kiểu dữ liệu.

Ví dụ: Tạo 5 số chia đều từ 0 đến 1.

```
>>> linArr = np.linspace(0, 1, 5)
# tao 5 so chia deu [0.0, 0.25, 0.5, 0.75, 1.0] giua 0 va 1
# ly tuong de tao mau deu trong mo hinh hoa toan hoc hoac ve bieu
    do
>>> print(linArr)
# hien thi mang ket qua: [0.   0.25 0.5  0.75 1.  ]
```

2.7 np.random

Chức năng: Tạo dữ liệu ngẫu nhiên, một công cụ quan trọng trong mô phỏng, học máy, và khởi tạo trọng số cho các mô hình AI.
Các phương thức con:

- `np.random.rand(shape)`: Số ngẫu nhiên trong `[0,1)`.
- `np.random.randn(shape)`: Số ngẫu nhiên theo phân phối chuẩn (mean=0, std=1).
- `np.random.randint(low, high, size)`: Số nguyên ngẫu nhiên trong `[low, high)`.

Tham số:

- **shape/size:** Kích thước mảng.
- **low, high:** Khoảng giá trị cho `randint`.

Ứng dụng: Mô phỏng dữ liệu, chia tập train/test trong học máy.
Ví dụ: Sinh dữ liệu ngẫu nhiên.

```
>>> randArr = np.random.rand(2, 3)
# tao mang 2x3 voi cac gia tri ngau nhien phan phoi deu giua 0 va
    1
# thuong dung de khoi tao trong so trong mang nho dan hoac mo
   phong Monte Carlo
>>> normArr = np.random.randn(5)
# tao 5 so ngau nhien tu phan phoi chuan (mean=0, std=1)
# huu ich de tao du lieu gia mo phong tieng noi thuc te
>>> intArr = np.random.randint(1, 10, 4)
# tao 4 so nguyen ngau nhien giua 1 va 9 (khong ke 10)
# ap dung trong thuat toan ngau nhien hoac phat trien game
>>> print(randArr, "\n", normArr, "\n", intArr)
# hien thi ket qua ngau nhien cua ba mang
```

3 Thuộc tính của mảng

3.1 shape

Chức năng: Trả về một tuple biểu diễn kích thước của mảng (số hàng, số cột, v.v.), giúp kiểm tra cấu trúc trước khi thực hiện các phép biến đổi hoặc tính toán.
Cú pháp: `a.shape`

Ứng dụng: Kiểm tra kích thước mảng trong các tác vụ tiền xử lý dữ liệu.
Ví dụ: Kiểm tra kích thước ma trận 3x4.

```
>>> a = np.zeros((3, 4))
# khoi tao ma tran 3x4 day so 0
# diem bat dau thuong dung cho cac phep tinh ma tran trong dai so
    tuyen tinh
>>> print(a.shape)
# tra ve kich thuoc mang duoi dang tuple (3, 4), chi 3 hang va 4
   cot
```

3.2 ndim

Chức năng: Trả về số chiều của mảng (1D, 2D, 3D, v.v.), hỗ trợ phân biệt loại mảng để áp dụng các phép xử lý phù hợp.
Cú pháp: `a.ndim`
Ứng dụng: Xác định số chiều trong các tác vụ đa chiều như xử lý hình ảnh.
Ví dụ: Kiểm tra số chiều của ma trận 2D.

```
>>> a = np.array([[1, 2], [3, 4]])
# dinh nghia mang 2D voi 2 hang va 2 cot
# dai dien cho cau truc ma tran trong du lieu da chieu
>>> print(a.ndim)
# tra ve so chieu: 2, chi rang day la cau truc 2D
```

3.3 size

Chức năng: Trả về tổng số phần tử trong mảng, giúp đánh giá dung lượng dữ liệu trước khi xử lý.
Cú pháp: `a.size`
Ứng dụng: Kiểm tra số lượng phần tử trong các bài toán phân tích dữ liệu lớn.
Ví dụ: Đếm tổng số phần tử.

```
>>> a = np.ones((2, 3))
# tao ma tran 2x3 day so 1
# thuong dung lam co so de tich luy gia tri trong xu ly du lieu
>>> print(a.size)
# tra ve tong so phan tu: 6 (2 hang * 3 cot)
```

3.4 dtype

Chức năng: Trả về kiểu dữ liệu của các phần tử trong mảng, hỗ trợ kiểm tra hoặc ép kiểu dữ liệu trong quá trình xử lý.
Cú pháp: `a.dtype`
Ứng dụng: Đảm bảo tính nhất quán kiểu dữ liệu trong các phép tính số học.
Ví dụ: Kiểm tra kiểu dữ liệu của mảng số thực.

```
>>> arr = np.array([1.2, 3.4])
# tao mang voi cac gia tri thuc
```

```
# thuong dung cho du lieu lien tuc trong tinh toan khoa hoc
>>> print(arr.dtype)
# tra ve kieu du lieu: float64, chi doan do chinh xac 64-bit
```

4 Đánh chỉ số và cắt lát

4.1 Truy cập phần tử cụ thể

Chức năng: Lấy hoặc sửa đổi giá trị của phần tử tại vị trí xác định, một thao tác cơ bản trong việc thao tác dữ liệu.
Cú pháp: `a[i,j]` (cho mảng 2D).

Ứng dụng: Truy xuất hoặc cập nhật dữ liệu trong các tập dữ liệu đa chiều.
Ví dụ: Lấy phần tử tại hàng 0, cột 1.

```
>>> a = np.array([[1, 2, 3], [4, 5, 6]])
# dinh nghia ma tran 2x3 voi cac gia tri integer
# dai dien cho tap du lieu ma moi phan tu co the duoc truy cap
    rieng le
>>> print(a[0, 1])
# lay phan tu tai hang 0, cot 1, tra ve: 2
```

4.2 Truy xuất hàng/cột

Chức năng: Trích xuất hàng, cột hoặc vùng con của mảng bằng kỹ thuật cắt lát, hỗ trợ phân tích dữ liệu chi tiết.
Cú pháp:

- `a[:, j]`: Cột thứ `j`.
- `a[i, :]`: Hàng thứ `i`.
- `a[i:j, k:l]`: Vùng con từ hàng `i` đến `j-1`, cột `k` đến `l-1`.

Ứng dụng: Tách dữ liệu để phân tích hoặc huấn luyện mô hình.
Ví dụ: Lấy cột đầu và hai hàng đầu.

```
>>> a = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
# tao ma tran 3x3 dai dien cho tap du lieu nho
# huu ich de trich xuat tap con trong phan tich du lieu
>>> print(a[:, 0])
# trich xuat cot dau: [1 4 7], chon tat ca hang cho cot 0
>>> print(a[0:2, :])
# trich xuat hai hang dau: [[1 2 3], [4 5 6]], su dung cat lat
```

4.3 Lọc điều kiện

Chức năng: Lấy các phần tử thỏa mãn điều kiện logic, một kỹ thuật quan trọng trong việc lọc dữ liệu theo ngưỡng.

Cú pháp: `a[condition]`

Ứng dụng: Lọc dữ liệu trong các tác vụ phân tích thời tiết hoặc tài chính.
Ví dụ: Lọc nhiệt độ lớn hơn 30.

```
>>> temps = np.random.randint(20, 40, 10)
# tao mang 10 so nguyen ngau nhien giua 20 va 39
# mo phong du lieu doc nhiet do tu cam bien
>>> hotTemps = temps[temps > 30]
# loc mang chi giu nhung nhiet do tren 30 do
# phep tinh thong dung trong phan tich du lieu khi hau
>>> print(hotTemps)
# tra ve mang cac gia tri vuot qua 30, vi du: [32 35 38]
```

4.4 np.where

Chức năng: Trả về chỉ số của phần tử thỏa mãn điều kiện hoặc thay thế giá trị dựa trên điều kiện, hỗ trợ xử lý dữ liệu thiếu và tối ưu hóa.
Cú pháp: `np.where(condition[, x, y])`
Tham số:

- **condition:** Biểu thức logic.
- **x, y:** Giá trị thay thế khi điều kiện đúng (`x`) hoặc sai (`y`). Nếu không có `x, y`, trả về chỉ số.

Ứng dụng: Xử lý dữ liệu thiếu, thay thế giá trị, hoặc tìm vị trí trong phân tích dữ liệu lớn.
Ví dụ: Thay thế giá trị < 0 bằng 0.

```
>>> data = np.array([-1, 2, -3, 4])
# tao mang voi cac gia tri am va duong
# dai dien cho tap du lieu can duoc chuan hoa
>>> result = np.where(data < 0, 0, data)
# thay the tat ca gia tri am bang 0, giu nguyen gia tri duong
# ky thuat dung de bao dam du lieu khong am trong tai chinh
>>> print(result)
# tra ve mang da sua: [0 2 0 4]
>>> indices = np.where(data < 0)
# tra ve chi so noi dieu kien (data < 0) dung
# huu ich de xac dinh vi tri bat thuong trong tap du lieu
>>> print(indices)
# tra ve: [0 2] (chi so cua -1 va -3)
```

4.5 np.isnan, np.isinf

Chức năng: Kiểm tra giá trị thiếu (`np.isnan`) hoặc vô cực (`np.isinf`), rất cần thiết trong việc làm sạch dữ liệu thực tế.
Cú pháp: `np.isnan(a)`, `np.isinf(a)`

Ứng dụng: Xử lý dữ liệu thiếu hoặc lỗi trong các tập dữ liệu lớn từ các nguồn không đồng nhất.
Ví dụ: Kiểm tra và loại bỏ giá trị `NaN`.

```
>>> data = np.array([1, np.nan, 3, np.inf, 5])
# tao mang voi gia tri NaN (thieu) va vo cuc
# mo phong du lieu thuc te co loi co the xay ra
>>> nanMask = np.isnan(data)
# tao bo loc boolean danh dau gia tri NaN
# can thiet de lam sach du lieu trong phan tich thong ke
>>> infMask = np.isinf(data)
# tao bo loc boolean danh dau gia tri vo cuc
# huu ich de phat hien tran so trong tinh toan so hoc
>>> print("NaN:", nanMask)
# tra ve: [False  True False False False]
>>> print("Inf:", infMask)
# tra ve: [False False False  True False]
>>> cleanData = data[~np.isnan(data) & ~np.isinf(data)]
# loc bo ca gia tri NaN va vo cuc
# buoc tien xu ly chuan trong ong day hoc may
>>> print(cleanData)
# tra ve: [1. 3. 5.]
```

4.6 np.clip

Chức năng: Giới hạn giá trị trong mảng trong khoảng `[a_min, a_max]`, giúp loại bỏ các giá trị ngoại lai trong dữ liệu.
Cú pháp: `np.clip(a, a_min, a_max)`
Tham số:

- **a:** Mảng đầu vào.

- $\mathbf{a}_m in, a_m ax$: $Giihndivtrn$. **Ứng dụng:** Chuẩn hóa dữ liệu trong xử lý tín hiệu hoặc hình ảnh.
Ví dụ: Giới hạn giá trị trong [0, 10].

```
>>> data = np.array([15, 5, -3, 12])
# tao mang voi cac gia tri ngoai khoang mong muon [0, 10]
# dai dien cho du lieu cam bien chua duoc chuan hoa
>>> clippedData = np.clip(data, 0, 10)
# gioi han tat ca gia tri trong khoang [0, 10], cat bo cuc tri
# ky thuat dung de tranh ngoai lai trong xu ly anh hoac tin hieu
>>> print(clippedData)
# tra ve: [10  5  0 10]
```

5 Các phép toán và thống kê

5.1 np.sum

Chức năng: Tính tổng các phần tử theo trục hoặc toàn mảng, một phép toán cơ bản trong phân tích dữ liệu số.
Cú pháp: `np.sum(a, axis=None, dtype=None)`
Tham số:

- **a:** Mảng đầu vào.
- **axis:** Trục tính tổng (`None`: toàn bộ, `0`: cột, `1`: hàng).
- **dtype:** Kiểu dữ liệu kết quả.

Ứng dụng: Tính tổng doanh thu hoặc điểm số trong các tập dữ liệu kinh doanh.
Ví dụ: Tính tổng doanh thu theo ngày và tổng cộng.

```
>>> salesData = np.array([[100, 200], [300, 400]])
# dinh nghia ma tran 2x2 dai dien cho du lieu ban hang hai ngay va hai san pham
>>> print(np.sum(salesData, axis=0))
# tinh tong theo cot (tong doc), tra ve [400 600]
# huu ich de tong hop tong ngay tren cac san pham
>>> print(np.sum(salesData))
# tinh tong tat ca phan tu trong ma tran, tra ve 1000
# phep tinh thong dung de tinh tong doanh thu tong the
```

5.2 np.mean, np.std

Chức năng: Tính trung bình (`np.mean`) và độ lệch chuẩn (`np.std`), các chỉ số thống kê quan trọng trong phân tích dữ liệu.
Cú pháp: `np.mean(a, axis=None)`, `np.std(a, axis=None)`
Ứng dụng: Chuẩn hóa z-score và phân tích độ biến thiên trong học máy.
Ví dụ: Chuẩn hóa z-score dữ liệu ngẫu nhiên.

```
>>> xData = np.random.randn(1000)
# tao 1000 so ngau nhien tu phan phoi chuan
# mo phong tap du lieu lon cho phan tich thong ke
>>> meanVal = np.mean(xData)
# tinh gia tri trung binh cua tap du lieu
# do luong khuynh huong trung tam trong thong ke mo ta
>>> stdVal = np.std(xData)
# tinh do lech chuan, chi ro do phan tan cua du lieu
# lien quan den chuan hoa trong hoc may
>>> zScores = (xData - meanVal) / stdVal
# chuan hoa du lieu thanh z-score voi trung binh ~0 va do lech chuan ~1
# buoc tien xu ly cho nhieu mo hinh thong ke
>>> print(zScores.mean(), zScores.std())
# kiem tra chuan hoa, du kien gia tri gan 0 va 1
```

5.3 np.min / np.max

Chức năng: Tìm giá trị nhỏ nhất (`np.min`) hoặc lớn nhất (`np.max`) trong mảng, hỗ trợ xác định phạm vi dữ liệu.
Cú pháp: `np.min(a, axis=None)`, `np.max(a, axis=None)`
Ứng dụng: Xác định cực trị như nhiệt độ cao nhất hoặc giá cổ phiếu thấp nhất.
Ví dụ: Tìm nhiệt độ thấp nhất và cao nhất.

```
>>> temps = np.array([25, 30, 28, 32, 27])
# tao mang du lieu doc nhiet do bang Celsius
# dai dien cho tap du lieu thoi gian nho
>>> print(np.min(temps), np.max(temps))
# tra ve nhiet do thap nhat va cao nhat: 25 32
# huu ich de xac dinh khoang trong giam sat moi truong
```

5.4 np.median, np.percentile

Chức năng: Tính trung vị (`np.median`) và phần trăm (`np.percentile`), cung cấp các chỉ số phân vị quan trọng trong thống kê.
Cú pháp: `np.median(a, axis=None)`, `np.percentile(a, q, axis=None)`
Tham số:

- **q:** Phần trăm (ví dụ: 90 cho percentile 90).

Ứng dụng: Phân tích phân phối và xác định ngưỡng bất thường trong dữ liệu.
Ví dụ: Tính trung vị và percentile 90 của doanh thu.

```
>>> revData = np.array([120, 100, 150, 130, 170])
# dinh nghia mang du lieu doanh thu cua mot doanh nghiep nho
>>> medianVal = np.median(revData)
# tinh gia tri trung vi (130), do luong trung tam manh me
>>> p90Val = np.percentile(revData, 90)
# tinh percentile 90 (164), chi nguong 10% doanh thu cao nhat
>>> print(medianVal, p90Val)
# tra ve: 130 164, huu ich de phat hien ngoai lai trong phan tich
    tai chinh
```

5.5 np.corrcoef

Chức năng: Tính hệ số tương quan Pearson giữa các biến, hỗ trợ phân tích mối quan hệ tuyến tính trong dữ liệu.
Cú pháp: `np.corrcoef(x, y=None)`
Tham số:

- **x:** Mảng 1D hoặc 2D (hàng là biến, cột là quan sát).
- **y:** Mảng 1D bổ sung (tùy chọn).

Ứng dụng: Đánh giá mối quan hệ giữa các đặc trưng trong học máy.
Ví dụ: Tính tương quan giữa doanh thu và chi phí quảng cáo.

```
>>> salesData = np.array([100, 120, 150, 130, 170])
# dai dien cho du lieu doanh thu nam trong nam ky
>>> adsData = np.array([50, 60, 80, 70, 90])
# dai dien cho chi phi quang cao trong cung ky
>>> corrMatrix = np.corrcoef(salesData, adsData)
# tinh ma tran he so tuong quan Pearson
# danh gia moi quan he tuyen tinh giua doanh thu va quang cao
```

```
>>> print(corrMatrix)
# tra ve ma tran 2x2, voi duong cheo 1 va gia tri tuong quan
    ngoai duong cheo
```

5.6 np.cov

Chức năng: Tính ma trận hiệp phương sai giữa các biến, hỗ trợ phân tích độ biến thiên trong các tập dữ liệu đa biến.
Cú pháp: `np.cov(m, bias=False)`
Tham số:

- **m:** Mảng 2D (hàng là biến, cột là quan sát).
- **bias:** Giá trị Boolean, chuẩn hóa theo N (True) hoặc N-1 (False).

Ứng dụng: Phân tích độ biến thiên giữa các đặc trưng trong học máy.
Ví dụ: Tính hiệp phương sai giữa doanh thu và chi phí.

```
>>> salesData = np.array([100, 120, 150, 130, 170])
# chua du lieu doanh thu cho nam quan sat
>>> adsData = np.array([50, 60, 80, 70, 90])
# chua du lieu chi phi quang cao cho cung ky
>>> covMatrix = np.cov([salesData, adsData])
# tinh ma tran hiep phuong sai, do luong do bien dong giua doanh
    thu va quang cao
# buoc quan trong trong phan tich thong ke da bien
>>> print(covMatrix)
# tra ve ma tran 2x2 voi phuong sai tren duong cheo va hiep
    phuong sai ngoai duong cheo
```

5.7 np.histogram

Chức năng: Tính biểu đồ tần suất của dữ liệu, một công cụ quan trọng trong việc hình dung phân phối dữ liệu.
Cú pháp: `np.histogram(a, bins=10, range=None, density=False)`
Tham số:

- **a:** Mảng đầu vào.
- **bins:** Số lượng hoặc danh sách các khoảng.
- **range:** Tuple (min, max) của dữ liệu.
- **density:** Giá trị Boolean, chuẩn hóa thành xác suất.

Ứng dụng: Phân tích phân phối dữ liệu trong các nghiên cứu thống kê.
Ví dụ: Tạo histogram cho dữ liệu ngẫu nhiên.

```
>>> data = np.random.randn(1000)
# tao 1000 mau ngau nhien tu phan phoi chuan
# dai dien cho tap du lieu lon de phan tich phan phoi
```

```
>>> histData, binsData = np.histogram(data, bins=10)
# tinh histogram voi 10 khoang, dem so lan xuat hien trong moi khoang
# cong cu co ban de hieu phan phoi du lieu
>>> print("Frequencies:", histData)
# tra ve so lan xuat hien trong moi khoang
>>> print("Bins:", binsData)
# tra ve canh khoang dinh nghia histogram
```

6 Thao tác hình dạng và Reshape

6.1 reshape

Chức năng: Thay đổi hình dạng của mảng mà không làm thay đổi dữ liệu, hỗ trợ chuẩn bị dữ liệu cho các mô hình học máy.
Cú pháp: `a.reshape(new_shape)`
Tham số:

- **new**$_s$*hape* : $Tuplehocsnguyn$(`-1`$tsuyra$). **Ứng dụng:** Biến đổi dữ liệu để phù hợp với các lớp đầu vào của mạng nơ-ron.
Ví dụ: Chuyển vector thành ma trận 2x3.

```
>>> a = np.arange(6)
# tao mang 1D [0, 1, 2, 3, 4, 5] su dung arange
# day so thuong dung lam co so de thay doi hinh dang
>>> print(a.reshape((2, 3)))
# thay doi mang thanh ma tran 2x3 [[0 1 2], [3 4 5]]
# bien doi thong dung de dieu chinh du lieu voi lop dau vao mo hinh
```

6.2 flatten

Chức năng: Chuyển mảng đa chiều thành mảng 1D, hỗ trợ chuẩn bị dữ liệu cho các thuật toán yêu cầu đầu vào vector.
Cú pháp: `a.flatten()`
Ứng dụng: Chuẩn bị dữ liệu đầu vào cho các mô hình học máy đơn giản.
Ví dụ: Làm phẳng ma trận 2D.

```
>>> a = np.array([[1, 2], [3, 4]])
# dinh nghia ma tran 2x2
# dai dien cho tap du lieu nho can duoc tuyen tinh hoa
>>> print(a.flatten())
# chuyen ma tran thanh mang 1D [1 2 3 4]
# huu ich de dua vao thuat toan can dau vao vector
```

6.3 transpose (T)

Chức năng: Chuyển vị ma trận bằng cách đổi hàng thành cột, một thao tác quan trọng trong đại số tuyến tính.

Cú pháp: `a.T` hoặc `np.transpose(a)`
Ứng dụng: Điều chỉnh ma trận trong các bài toán học máy hoặc xử lý hình ảnh.
Ví dụ: Chuyển vị ma trận 2x3.

```
>>> a = np.array([[1, 2, 3], [4, 5, 6]])
# tao ma tran 2x3
# thuong dung de dai dien du lieu voi 2 mau va 3 dac trung
>>> print(a.T)
# chuyen vi ma tran thanh 3x2 [[1 4], [2 5], [3 6]]
# bien doi can thiet cho cac phep tinh ma tran hoac canh chinh
    dac trung
```

6.4 np.concatenate

Chức năng: Ghép nhiều mảng thành một theo trục được chỉ định, hỗ trợ kết hợp dữ liệu từ nhiều nguồn.
Cú pháp: `np.concatenate((a1, a2, ...), axis=0)`
Tham số:

- **(a1, a2, ...):** Tuple các mảng cần ghép.
- **axis:** Trục ghép (`0`: hàng, `1`: cột).

Ứng dụng: Kết hợp tập dữ liệu trong các dự án tiền xử lý dữ liệu lớn.
Ví dụ: Ghép hai mảng theo hàng.

```
>>> a1 = np.array([[1, 2], [3, 4]])
# dinh nghia ma tran 2x2, dai dien cho tap con du lieu
>>> a2 = np.array([[5, 6]])
# dinh nghia ma tran 1x2, tap con khac de ghep lai
>>> print(np.concatenate((a1, a2), axis=0))
# ghep theo hang, tra ve [[1 2], [3 4], [5 6]]
# phuong phap de hop nhat du lieu theo chieu doc trong tien xu ly
```

6.5 np.vstack, np.hstack

Chức năng: Ghép mảng theo chiều dọc (`vstack`) hoặc chiều ngang (`hstack`), hỗ trợ tổ chức dữ liệu theo cấu trúc mong muốn.
Cú pháp: `np.vstack((a1, a2, ...))`, `np.hstack((a1, a2, ...))`
Ứng dụng: Kết hợp dữ liệu trong các tác vụ tiền xử lý hoặc phân tích đa chiều.
Ví dụ: Ghép theo chiều dọc và ngang.

```
>>> a1 = np.array([[1, 2], [3, 4]])
# dinh nghia ma tran 2x2, phan cua tap du lieu lon hon
>>> a2 = np.array([[5, 6]])
# dinh nghia ma tran 1x2, phan khac de ghep vao
>>> print(np.vstack((a1, a2)))
# ghep theo chieu doc, tra ve [[1 2], [3 4], [5 6]]
# huu ich de them hang moi trong du lieu thoi gian
>>> print(np.hstack((a1, a2.T)))
```

```
# ghep theo chieu ngang sau khi chuyen vi a2
# tra ve [[1 2 5], [3 4 6]], ly tuong de ket hop dac trung
```

6.6 np.unique

Chức năng: Trả về các giá trị duy nhất trong mảng, hỗ trợ phân tích tần suất và loại bỏ trùng lặp trong dữ liệu.
Cú pháp: `np.unique(ar, returnCounts=False)`
Tham số:

- **ar:** Mảng đầu vào.
- **returnCounts:** Giá trị Boolean, trả về số lần xuất hiện.

Ứng dụng: Phân tích dữ liệu phân loại hoặc chuẩn bị dữ liệu cho mô hình.
Ví dụ: Tìm giá trị duy nhất và tần suất.

```
>>> data = np.array([1, 2, 2, 3, 1, 4])
# tao mang voi cac gia tri trung lap
# mo phong tap du lieu do do nhieu lan
>>> uniqueVals, counts = np.unique(data, returnCounts=True)
# xac dinh cac gia tri duy nhat va tan suat xuat hien
# buoc quan trong trong phan tich du lieu danh muc
>>> print(uniqueVals, counts)
# tra ve: [1 2 3 4] [2 2 1 1], chi so lan xuat hien cua moi gia
    tri
```

6.7 np.vectorize

Chức năng: Áp dụng hàm tùy chỉnh lên từng phần tử của mảng, tăng cường khả năng xử lý dữ liệu linh hoạt.
Cú pháp: `np.vectorize(pyfunc, otypes=None)`
Tham số:

- **pyfunc:** Hàm Python cần áp dụng.
- **otypes:** Kiểu dữ liệu đầu ra.

Ứng dụng: Tùy chỉnh xử lý dữ liệu trong các dự án học máy phức tạp.
Ví dụ: Áp dụng hàm bình phương.

```
>>> def squareFunc(x):
# dinh nghia ham tinh binh phuong cua mot so
# bien doi don gian de trinh bay
    return x**2
>>> vecSquare = np.vectorize(squareFunc)
# chuyen ham thanh vector de ap dung len moi phan tu mang
# tang hieu suat so voi vong lap Python tren du lieu lon
>>> data = np.array([1, 2, 3])
# tao mang mau de kiem tra ham vector hoa
```

```
>>> print(vecSquare(data))
# ap dung ham binh phuong, tra ve [1 4 9]
# huu ich cho cac bien doi phan tu trong tien xu ly du lieu
```

7 Sắp xếp và Tìm kiếm

7.1 np.sort

Chức năng: Sắp xếp mảng theo thứ tự tăng dần, hỗ trợ tổ chức dữ liệu trong các tác vụ phân tích.
Cú pháp: `np.sort(a, axis=-1, kind='quicksort')`
Tham số:

- **axis:** Trục sắp xếp (`-1`: trục cuối, `None`: làm phẳng).
- **kind:** Thuật toán (`'quicksort'`, `'mergesort'`, `'heapsort'`).

Ứng dụng: Sắp xếp dữ liệu số hoặc chuỗi trong các bài toán tối ưu hóa.
Ví dụ: Sắp xếp doanh thu tăng dần.

```
>>> revData = np.array([120, 100, 150, 130])
# dinh nghia mang du lieu doanh thu
# dai dien cho du lieu tai chinh can sap xep
>>> print(np.sort(revData))
# sap xep mang theo thu tu tang dan: [100 120 130 150]
# phep tinh thong dung de xep hang hoac phan tich xu huong
```

7.2 np.argsort

Chức năng: Trả về chỉ số của phần tử khi sắp xếp tăng dần, hỗ trợ sắp xếp đồng bộ nhiều mảng.
Cú pháp: `np.argsort(a, axis=-1)`
Ứng dụng: Sắp xếp liên kết dữ liệu trong các tập dữ liệu phức tạp.
Ví dụ: Tìm chỉ số của doanh thu khi sắp xếp.

```
>>> revData = np.array([120, 100, 150, 130])
# chua du lieu doanh thu theo thu tu ban dau
# can chi so sap xep de dong bo du lieu lien quan
>>> print(np.argsort(revData))
# tra ve chi so [1 0 3 2] de sap xep mang
# giup dong bo sap xep cac mang lien ket
```

7.3 np.argmax / np.argmin

Chức năng: Trả về chỉ số của phần tử lớn nhất (`np.argmax`) hoặc nhỏ nhất (`np.argmin`), hỗ trợ tìm vị trí cực trị.
Cú pháp: `np.argmax(a, axis=None)`, `np.argmin(a, axis=None)`
Ứng dụng: Xác định vị trí cực trị trong các chuỗi thời gian hoặc dữ liệu kinh doanh.
Ví dụ: Tìm ngày có doanh thu cao nhất.

```
>>> revData = np.array([120, 100, 150, 130])
# dai dien cho du lieu doanh thu trong bon ngay
# vi tri gia tri cao nhat la muc tieu phan tich
>>> print(np.argmax(revData))
# tra ve chi so 2, chi vi tri cua gia tri lon nhat 150
# huu ich de xac dinh ngay hieu suat cao nhat
```

8 Broadcasting và Đại số tuyến tính

8.1 Broadcasting

Chức năng: Thực hiện phép toán trên các mảng có kích thước khác nhau bằng cách tự động mở rộng kích thước, tối ưu hóa tính toán trong NumPy.

Ứng dụng: Chuẩn hóa dữ liệu hoặc áp dụng phép toán trên toàn mảng mà không cần vòng lặp.

Ví dụ: Chuẩn hóa ma trận bằng cách trừ trung bình mỗi cột.

```
>>> data = np.array([[1, 2], [3, 4], [5, 6]])
# dinh nghia ma tran 3x2 dai dien cho tap du lieu voi hai dac
    trung
>>> meanVals = np.mean(data, axis=0)
# tinh trung binh theo cot [3 4]
# buoc de dua du lieu ve trung tam trong chuan hoa thong ke
>>> normalizedData = data - meanVals
# ap dung broadcasting de tru trung binh khoi moi hang
# tra ve ma tran da chuan hoa cho phan tich tiep theo
>>> print(normalizedData)
# tra ve: [[-2 -2], [0 0], [2 2]], chi do lech tu trung binh cot
```

8.2 np.dot

Chức năng: Thực hiện nhân ma trận hoặc tích vô hướng, một phép toán cốt lõi trong đại số tuyến tính và học máy.

Cú pháp: `np.dot(a, b)`

Ứng dụng: Tính toán trong mạng nơ-ron hoặc biến đổi hình học.

Ví dụ: Nhân hai ma trận 2x2.

```
>>> A = np.array([[1, 2], [3, 4]])
# dinh nghia ma tran 2x2 A, ma tran he so trong he phuong trinh
    tuyen tinh
>>> B = np.array([[5, 6], [7, 8]])
# dinh nghia ma tran 2x2 B, doi tuong thu hai de nhan
>>> print(np.dot(A, B))
# thuc hien nhan ma tran, tra ve [[19 22], [43 50]]
# phep tinh trung tam trong bien doi tuyen tinh va mang nho dan
```

8.3 np.linalg

Chức năng: Cung cấp các hàm đại số tuyến tính như nghịch đảo ma trận, định thức, giá trị riêng, và giải hệ phương trình, hỗ trợ các bài toán toán học nâng cao.
Cú pháp:

- `np.linalg.inv(A)`: Nghịch đảo ma trận (yêu cầu ma trận vuông và khả nghịch).
- `np.linalg.det(A)`: Tính định thức ma trận (định lượng khả nghịch, 0 nếu không khả nghịch).
- `np.linalg.eig(A)`: Tính giá trị riêng và vector riêng (phân tích phổ ma trận).
- `np.linalg.solve(A, b)`: Giải hệ phương trình tuyến tính `Ax = b` (A phải khả nghịch).

Ứng dụng: Giải hệ phương trình, phân tích thành phần chính trong học máy.
Ví dụ: Giải hệ phương trình `Ax = b`.

```
>>> A = np.array([[3, 1], [2, 4]])
# dinh nghia ma tran he so 2x2 A cho he phuong trinh tuyen tinh
# dai dien cho mo hinh vat ly hoac kinh te
>>> b = np.array([7, 10])
# dinh nghia vector 2x1 b lam ben phai cua phuong trinh Ax = b
>>> x = np.linalg.solve(A, b)
# giai he Ax = b, voi A la ma tran kha nghich
# phuong phap dung trong toi uu hoa va mo phong vat ly
>>> print(x)
# tra ve giai phap: [2. 1.], cac gia tri x thoa man Ax = b
>>> evals, evecs = np.linalg.eig(A)
# tinh gia tri rieng va vector rieng cua A
# can thiet cho phan tich pho trong phan tich thanh phan chinh
>>> print(evals)
# tra ve gia tri rieng, chi cac he so phong to cua ma tran
```

8.4 np.einsum

Chức năng: Thực hiện phép toán tensor hiệu quả bằng ký hiệu Einstein, cho phép định nghĩa tùy chỉnh cách các chiều được kết hợp, tối ưu hóa tính toán phức tạp.
Cú pháp: `np.einsum(subscripts, *operands)`
Tham số:

- **subscripts:** Chuỗi ký hiệu (ví dụ: `'ij,jk->ik'` cho nhân ma trận, `'ij,ij->'` cho tổng tích từng phần tử), mô tả cách các chiều được ánh xạ.
- **operands:** Các mảng đầu vào tham gia phép toán.

Ứng dụng: Tính toán tensor trong học sâu, đặc biệt hữu ích với dữ liệu đa chiều.
Ví dụ: Nhân ma trận và tính tổng theo hàng.

```
>>> A = np.array([[1, 2], [3, 4]])
# dinh nghia ma tran 2x2 A, mau tensor de trinh bay
>>> B = np.array([[5, 6], [7, 8]])
```

```
# dinh nghia ma tran 2x2 B, doi tuong khac cho phep toan
>>> result = np.einsum('ij,ij->', A, B)
# su dung ky hieu Einstein 'ij,ij->' de tinh tong tich tung phan tu
# tuong duong voi tich vo huong cua mang da de, tra ve 70 (1*5 + 2*6 + 3*7 + 4*8)
# phuong phap toi uu hoa cho cac phep rut gon tensor
>>> print("Sum of products:", result)
# tra ve: 70, ket qua so cua tong tich tung phan tu
>>> sumRows = np.einsum('ij->i', A)
# tinh tong phan tu theo hang su dung ky hieu 'ij->i'
# giam mang 2D xuong mang 1D voi tong hang
>>> print(sumRows)
# tra ve: [3 7], tong cua moi hang, huu ich de tong hop dac trung
```

9 Xử lý dữ liệu lớn

9.1 np.save, np.load

Chức năng: Lưu (`np.save`) và tải (`np.load`) mảng vào/từ file định dạng .npy, hỗ trợ lưu trữ dữ liệu số hiệu quả.
Cú pháp: `np.save(file, arr)`, `np.load(file)`
Tham số:

- **file:** Tên file hoặc đối tượng file.
- **arr:** Mảng cần lưu.

Ứng dụng: Lưu trữ và tái sử dụng dữ liệu lớn trong các dự án học máy hoặc phân tích dữ liệu.
Ví dụ: Lưu và tải mảng.

```
>>> data = np.array([[1, 2], [3, 4]])
# tao ma tran 2x2, tap du lieu nho de luu tru
>>> np.save('dataFile.npy', data)
# luu mang NumPy vao file nhi phan .npy
# dang hieu qua de luu tru du lieu so lon
>>> loadedData = np.load('dataFile.npy')
# tai mang tu file .npy da luu
# giup lay lai du lieu de phan tich hoac dao mo hinh tiep theo
>>> print(loadedData)
# hien thi ma tran da tai lai: [[1 2], [3 4]]
```

10 Ví dụ thực hành từ Kaggle

Phần này cung cấp các bài tập thực hành lấy cảm hứng từ nền tảng Kaggle, nhằm giúp học viên áp dụng kiến thức NumPy vào các tình huống thực tế trong Khoa học Dữ liệu và AI.

10.1 Bài 1: Chuẩn hóa dữ liệu nhiệt độ

Bài toán: Chuẩn hóa danh sách nhiệt độ về z-score để chuẩn bị dữ liệu cho phân tích thống kê.
Lời giải:

```
>>> import numpy as np
>>> temps = np.array([25, 28, 30, 27, 29, 31, 26])
# dinh nghia mang doc nhiet do bang Celsius
# dai dien cho du lieu thoi tiet trong mot tuan
>>> meanTemp = np.mean(temps)
# tinh trung binh nhiet do tren tap du lieu
# do luong khuynh huong trung tam de chuan hoa
>>> stdTemp = np.std(temps)
# tinh do lech chuan, chi do bien dong nhiet do
>>> zScores = (temps - meanTemp) / stdTemp
# chuan hoa nhiet do thanh z-score
# bien doi du lieu ve phan phoi chuan de phan tich thong ke
>>> print("Original temperatures:", temps)
>>> print("Z-scores:", zScores)
# tra ve nhiet do ban dau va z-score cua chung
# voi trung binh ~0 va do lech chuan ~1
```

10.2 Bài 2: Tìm ngày doanh thu cao nhất

Bài toán: Xác định ngày có doanh thu cao nhất và thấp nhất trong một chuỗi dữ liệu kinh doanh.
Lời giải:

```
>>> salesData = np.array([120, 100, 150, 130, 170, 110, 140])
# dai dien cho du lieu doanh thu trong bay ngay
# tap du lieu thoi gian cho phan tich kinh doanh
>>> maxDay = np.argmax(salesData)
# xac dinh chi so cua gia tri doanh thu cao nhat
# chi ngay co hieu suat cao nhat
>>> minDay = np.argmin(salesData)
# xac dinh chi so cua gia tri doanh thu thap nhat
# chi ngay co hieu suat thap nhat
>>> print("Revenue:", salesData)
>>> print("Highest day:", maxDay, "with revenue", salesData[
    maxDay])
# tra ve ngay va doanh thu cao nhat, vi du: "Highest day: 4 with
    revenue 170"
>>> print("Lowest day:", minDay, "with revenue", salesData[minDay
    ])
# tra ve ngay va doanh thu thap nhat, vi du: "Lowest day: 1 with
    revenue 100"
```

10.3 Bài 3: Tính toán ma trận

Bài toán: Tính nghịch đảo của ma trận 2x2 và nhân với vector để giải hệ phương trình tuyến tính.
Lời giải:

```
>>> A = np.array([[4, 7], [2, 6]])
# dinh nghia ma tran he so 2x2 A cho he phuong trinh tuyen tinh
# dai dien cho mo hinh vat ly hoac kinh te
>>> b = np.array([8, 10])
# dinh nghia vector 2x1 b lam ben phai cua phuong trinh Ax = b
>>> AInv = np.linalg.inv(A)
# tinh nghich dao cua ma tran A
# yeu cau A la ma tran vuong va kha nghich, buoc de giai he
>>> result = np.dot(AInv, b)
# nhan nghich dao A voi vector b de giai cho x trong Ax = b
# phuong phap thong dung trong bai toan toi uu hoa
>>> print("Matrix A:\n", A)
>>> print("Inverse A:\n", AInv)
>>> print("Result:", result)
# tra ve ma tran ban dau, nghich dao va giai phap [2. 1.]
```

10.4 Bài 4: Phân tích dữ liệu phân loại

Bài toán: Đếm tần suất các giá trị duy nhất trong tập dữ liệu phân loại để chuẩn bị cho mô hình học máy.
Lời giải:

```
>>> categories = np.array(['A', 'B', 'A', 'C', 'B', 'A', 'C', 'C'
    ])
# tao mang du lieu danh muc voi nhan trung lap
# mo phong tap du lieu nhu phan doan khach hang
>>> uniqueCats, counts = np.unique(categories, returnCounts=True)
# xac dinh cac danh muc duy nhat va tan suat xuat hien
# buoc tien xu ly de tao dac trung trong hoc may
>>> print("Categories:", uniqueCats)
>>> print("Frequencies:", counts)
# tra ve: Categories: ['A' 'B' 'C'] Frequencies: [3 2 3]
# chi phan phoi cac danh muc trong tap du lieu
```

10.5 Bài 5: Xử lý dữ liệu thiếu

Bài toán: Thay thế giá trị `NaN` bằng trung bình của mảng để xử lý dữ liệu không hoàn chỉnh.
Lời giải:

```
>>> data = np.array([1, np.nan, 3, 4, np.nan])
# tao mang voi gia tri thieu (NaN)
# mo phong tap du lieu thuc te co do mat mat
>>> meanVal = np.nanmean(data)
```

```
# tinh trung binh, bo qua gia tri NaN
# phuong phap manh me de xu ly du lieu thieu trong thong ke
>>> cleanedData = np.where(np.isnan(data), meanVal, data)
# thay the gia tri NaN bang trung binh tinh duoc
# ky thuat do de chuan bi du lieu cho phan tich
>>> print("Original data:", data)
>>> print("Cleaned data:", cleanedData)
# tra ve du lieu ban dau va du lieu da duoc do, vi du: Original:
    [1. nan 3. 4. nan], Cleaned: [1. 2.666 3. 4. 2.666]
```

10.6 Bài 6: Phân tích phân phối dữ liệu

Bài toán: Tạo histogram cho tập dữ liệu ngẫu nhiên và phân tích phân phối để đánh giá đặc điểm thống kê.
Lời giải:

```
>>> data = np.random.randn(1000)
# tao 1000 mau ngau nhien tu phan phoi chuan
# dai dien cho tap du lieu lon de phan tich phan phoi
>>> histData, binsData = np.histogram(data, bins=10, density=True
    )
# tinh histogram voi 10 khoang, chuan hoa thanh mat do xac suat
# cong cu thong ke de trinh bay phan phoi du lieu
>>> print("Normalized frequencies:", histData)
# tra ve tan suat chuan hoa cho moi khoang
# tong bang 1 do chuan hoa mat do
>>> print("Bins:", binsData)
# tra ve canh khoang dinh nghia cac khoang histogram
```

10.7 Bài 7: Tính toán tensor

Bài toán: Sử dụng `np.einsum` để tính tổng tích các phần tử của hai ma trận, minh họa tính toán nâng cao.
Lời giải:

```
>>> A = np.array([[1, 2], [3, 4]])
# dinh nghia ma tran 2x2 A, mau tensor de thuc hien phep toan
>>> B = np.array([[5, 6], [7, 8]])
# dinh nghia ma tran 2x2 B, doi tuong khac cho tinh toan
>>> result = np.einsum('ij,ij->', A, B)
# su dung ky hieu Einstein 'ij,ij->' de tinh tong tich tung phan
    tu
# tuong duong voi tich vo huong cua mang da de, tra ve 70 (1*5 +
    2*6 + 3*7 + 4*8)
# phuong phap toi uu hoa cho cac phep rut gon tensor trong hoc
    sau
>>> print("Sum of products:", result)
# tra ve: 70, ket qua so cua tong tich tung phan tu
```

Tài liệu Tham khảo

- NumPy Documentation: https://numpy.org/doc/stable/
- Học ứng dụng Kaggle: https://www.kaggle.com/learn/
- Scipy Lecture Notes: https://scipy-lectures.org/

Hướng dẫn Sử dụng cho Học viên

- Tài liệu này được thiết kế để học viên có thể tự học hoặc sử dụng trong các khóa đào tạo về Khoa học Dữ liệu và AI.
- Mỗi ví dụ đều đi kèm chú thích chi tiết, giúp học viên hiểu rõ ý nghĩa và ứng dụng thực tế của từng phương thức.
- Học viên được khuyến khích thực hành các bài tập thực tế và mở rộng bằng cách tích hợp với các thư viện khác như Pandas, Matplotlib hoặc Scikit-learn.
- Nếu gặp lỗi biên dịch hoặc cần hỗ trợ thêm, hãy liên hệ với người hướng dẫn hoặc tham khảo các tài liệu tham khảo được liệt kê.