Python cho Khoa học dữ liệu

Bài 18: Tổng quan về Thư viện Pandas

Hà Minh Tuấn
hmtuan@hcmus.edu.vn

Khoa Toán - Tin học
Trường Đại học Khoa học Tự nhiên, ĐHQG-HCM

Ngày 29 tháng 10 năm 2025

Nội dung bài học

Pandas là gì?

- Pandas là thư viện Python mạnh mẽ để xử lý và phân tích dữ liệu.
- Hai cấu trúc dữ liệu chính:
  - **Series**: mảng 1 chiều có nhãn.
  - **DataFrame**: bảng 2 chiều với hàng và cột có nhãn.
- Được sử dụng rộng rãi trong Khoa học dữ liệu, Tài chính, Y sinh, Xã hội học, v.v.

Tạo Series

```
import pandas as pd

s = pd.Series([10, 20, 30, 40],
              index=['a','b','c','d'])
print(s)
```

```
a    10
b    20
c    30
d    40
dtype: int64
```

Tạo DataFrame

```
data = {'Tên': ['An', 'Bình', 'Cường'],
        'Tuổi': [20, 21, 19],
        'Điểm': [8.5, 7.0, 9.0]}
df = pd.DataFrame(data)
print(df)
```

```
     Tên  Tuổi  Điểm
0     An    20   8.5
1   Bình    21   7.0
2  Cường    19   9.0
```

Chỉ số và chọn lọc

```python
print(df.loc[0])          # chọn theo nhãn
print(df.iloc[1])         # chọn theo chỉ số
print(df['Tên'])          # chọn 1 cột
```

```
Tên      An
Tuổi     20
Điểm    8.5
Name: 0, dtype: object

Tên      Bình
Tuổi       21
Điểm      7.0
Name: 1, dtype: object

0       An
1     Bình
2   Cường
```

Xử lý Missing Values

```
df2 = pd.DataFrame({
    'A': [1, 2, None],
    'B': [4, None, 6]
})
print(df2.fillna(0))      # thay NaN bằng 0
```

```
     A    B
0  1.0  4.0
1  2.0  0.0
2  0.0  6.0
```

Sắp xếp và lọc

```
print(df.sort_values(by='Điểm'))
print(df[df['Điểm'] > 8])
```

```
     Tên  Tuổi  Điểm
1   Bình    21   7.0
0     An    20   8.5
2  Cường    19   9.0

     Tên  Tuổi  Điểm
0     An    20   8.5
2  Cường    19   9.0
```

Thống kê mô tả

```
print(df.describe())
```

```
              Tuổi      Điểm
count    3.000000  3.000000
mean    20.000000  8.166667
std      1.000000  1.040835
min     19.000000  7.000000
25%     19.500000  7.750000
50%     20.000000  8.500000
75%     20.500000  8.750000
max     21.000000  9.000000
```

GroupBy

```
df3 = pd.DataFrame({
    'Lớp': ['A','A','B','B'],
    'Điểm': [8, 6, 9, 7]
})
print(df3.groupby('Lớp').mean())
```

```
      Điểm
Lớp
A      7.0
B      8.0
```

Merge DataFrame

```
dfA = pd.DataFrame({'ID':[1,2], 'Tên':['An','Bình']})
dfB = pd.DataFrame({'ID':[1,2], 'Điểm':[8.5,7.0]})

merged = pd.merge(dfA, dfB, on='ID')
print(merged)
```

```
   ID   Tên  Điểm
0   1    An   8.5
1   2  Bình   7.0
```

Xử lý dữ liệu thời gian

```
dates = pd.date_range('2025-01-01', periods=3)
ts = pd.Series([1,2,3], index=dates)
print(ts)
```

```
2025-01-01    1
2025-01-02    2
2025-01-03    3
Freq: D, dtype: int64
```

Áp dụng hàm với apply

```
print(df['Điểm'].apply(lambda x: x**2))
```

```
0    72.25
1    49.00
2    81.00
Name: Điểm, dtype: float64
```

Hiệu năng Pandas

- Pandas xây dựng trên NumPy, tối ưu cho dữ liệu dạng bảng.
- Nên tận dụng vectorization thay vì vòng lặp.
- Với dữ liệu rất lớn: dùng `chunksize`, `Dask` hoặc `Polars`.

pandas.DataFrame.pivot — Giới thiệu

- **Cú pháp:** `DataFrame.pivot(*, columns, index=<no_default>, values=<no_default>)`
- **Chức năng:** Trả về một `DataFrame` được **tái cấu trúc (reshaped)** dựa trên các giá trị của **index** và **columns** được chỉ định.
- Hàm này tạo ra **bảng trục (pivot table)** từ dữ liệu có cấu trúc dạng "dài", biến chúng thành dạng "rộng".
- **Lưu ý:** Hàm `pivot()` **không hỗ trợ tổng hợp dữ liệu (aggregation)**. Nếu có nhiều giá trị trùng lặp cho cùng một cặp `index/column`, sẽ sinh ra lỗi `ValueError`.

Tham số của `DataFrame.pivot`

- **columns:**
  - Chuỗi, đối tượng, hoặc danh sách tên cột.
  - Dùng để xác định các cột mới trong `DataFrame` kết quả.
- **index:**
  - Chuỗi, đối tượng, hoặc danh sách tên cột (tùy chọn).
  - Dùng để xác định chỉ mục (index) mới. Nếu không chỉ định, dùng index hiện có.
- **values:**
  - Chuỗi hoặc danh sách tên cột.
  - Xác định cột (hoặc các cột) chứa giá trị để đưa vào bảng kết quả.
  - Nếu không chỉ định, tất cả các cột còn lại sẽ được sử dụng và tạo **MultiIndex**.

Giá trị trả về và lỗi có thể gặp

- **Trả về:**
  - Một `DataFrame` đã được tái cấu trúc theo các giá trị `index`, `columns` và `values`.
- **Lỗi:**
  - `ValueError`: Xuất hiện khi có các tổ hợp `index / column` trùng lặp với nhiều giá trị.
  - Trong trường hợp này, nên dùng `DataFrame.pivot_table()` để tổng hợp dữ liệu.

Ví dụ cơ bản về pivot()

```
import pandas as pd
df = pd.DataFrame({
    'foo': ['one', 'one', 'one', 'two', 'two', 'two'],
    'bar': ['A', 'B', 'C', 'A', 'B', 'C'],
    'baz': [1, 2, 3, 4, 5, 6],
    'zoo': ['x', 'y', 'z', 'q', 'w', 't']
})
print(df)
```

	foo	bar	baz	zoo
0	one	A	1	x
1	one	B	2	y
2	one	C	3	z
3	two	A	4	q
4	two	B	5	w
5	two	C	6	t

Thực hiện pivot đơn giản

```
df.pivot(index='foo', columns='bar', values='baz')
```

```
bar  A   B   C
foo
one  1   2   3
two  4   5   6
```

```
df.pivot(index='foo', columns='bar')['baz']
```

```
bar  A   B   C
foo
one  1   2   3
two  4   5   6
```

Pivot với nhiều cột giá trị

```
df.pivot(index='foo', columns='bar', values=['baz', 'zoo'])
```

	baz			zoo		
bar	A	B	C	A	B	C
foo						
one	1	2	3	x	y	z
two	4	5	6	q	w	t

Pivot với nhiều cấp độ chỉ mục và cột

```
df = pd.DataFrame({
   "lev1": [1, 1, 1, 2, 2, 2],
   "lev2": [1, 1, 2, 1, 1, 2],
   "lev3": [1, 2, 1, 2, 1, 2],
   "lev4": [1, 2, 3, 4, 5, 6],
   "values": [0, 1, 2, 3, 4, 5]
})
df.pivot(index="lev1", columns=["lev2", "lev3"], values="values")
```

lev2	1		2	
lev3	1	2	1	2
lev1				
1	0.0	1.0	2.0	NaN
2	4.0	3.0	NaN	5.0

Pivot với nhiều chỉ mục phức hợp

```
df.pivot(index=["lev1", "lev2"], columns=["lev3"], values="values")
```

```
      lev3    1    2
lev1  lev2
   1     1  0.0  1.0
         2  2.0  NaN
   2     1  4.0  3.0
         2  NaN  5.0
```

Lỗi khi có giá trị trùng lặp

```
df = pd.DataFrame({
    "foo": ['one', 'one', 'two', 'two'],
    "bar": ['A', 'A', 'B', 'C'],
    "baz": [1, 2, 3, 4]
})
df.pivot(index='foo', columns='bar', values='baz')
```

```
ValueError: Index contains duplicate entries, cannot reshape
```

Ghi chú và liên kết tham khảo

- Để xử lý các trường hợp dữ liệu trùng lặp, nên dùng:
  - `DataFrame.pivot_table()` — cho phép tổng hợp dữ liệu.
  - `DataFrame.unstack()` — xoay trục dựa trên index thay vì cột.
  - `wide_to_long()` — chuyển dữ liệu dạng rộng thành dài.
- Xem thêm tài liệu về **hierarchical indexing** để hiểu cách hoạt động của các cấu trúc MultiIndex.