19. Ghép nối các cấu trúc dữ liệu pandas

Trong chuỗi các bài học tiếp theo sẽ đề cập đến kĩ thuật merging dataframe. Xuất phát từ yêu cầu thực tế là chúng ta cần đọc dữ liệu tồn tại dưới nhiều định dạng khác nhau hoặc (và) nhiều files khác nhau. Đọc dữ liệu từ nhiều files và nhiều định dạng qua các tools mà pandas cung cấp như: pd.read_csv(), pd.read_excel(), pd.read_html(), pd.read_json(), chúng ta sẽ nhận được nhiều dataframe và mục tiêu của chúng ta là cần ghép nối chúng để có được một dataframe mong muốn. Chúng ta sẽ dần làm rõ các vấn đề này trong bài hôm nay và những chuỗi bài tiếp theo.

Ghép nối các cấu trúc dữ liệu pandas (Concatenating pandas structure)

Pandas cung cấp hai phương thức:

1. pd.concat(). Linh hoạt và có thể làm tốt cả việc ghép nối theo hàng và cột.

2. .append(). Chỉ ghép nối theo hàng nhưng cách sử dụng linh hoạt dễ nhớ.

Phương thức concat thực hiện tất cả các hoạt động ghép nối trên một trục trong khi thực hiện logic như union hoặc intersection của các chỉ mục (nếu có) trên các trục khác. Lưu ý rằng tôi nói "nếu có" bởi vì chỉ có một trục ghép nối cho kiểu Series.

pd.concat(objs, axis=0, join='outer', join_axes=None, ignore_index=False, keys=None, levels=None, names=None, ve

Chúng ta sẽ làm rõ ý nghĩa của các tham số qua các ví dụ.

Từ khóa axis

Mặc định axis = 0, nghĩa là các dataframe sẽ được ghép chồng theo hàng (row).

Sử dụng 3 dataframes: df1, df2, df3.

```
>>> df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],'B': ['B0', 'B1', 'B2', 'B3'],'C': ['C0', 'C1', 'C2', 'C3'],'D': ['D0', 'D1', 'D2', 'D3']},index=[0, 1, 2, 3])
>>> df2 = pd.DataFrame({'A': ['A4', 'A5', 'A6', 'A7'],'B': ['B4', 'B5', 'B6', 'B7'],'C': ['C4', 'C5', 'C6', 'C7'],'D': ['D4', 'D5', 'D6', 'D7']},index=[0, 1, 2, 3])
>>> df3 = pd.DataFrame({'A': ['A8', 'A9', 'A10', 'A11'],'B': ['B8', 'B9', 'B10', 'B11'],'C': ['C8', 'C9', 'C10', 'C11'],'D': ['D8', 'D9', 'D10', 'D11']},index=[0, 2, 3, 4])
>>>
```

Cụ thể hình dạng df1, df2, df3 như sau:

```
>>> df1

  A  B  C  D

0 A0 B0 C0 D0

1 A1 B1 C1 D1

2 A2 B2 C2 D2

3 A3 B3 C3 D3

>>> df2

  A  B  C  D

0 A4 B4 C4 D4

1 A5 B5 C5 D5

2 A6 B6 C6 D6

3 A7 B7 C7 D7

>>> df3

   A   B   C   D

0  A8  B8  C8  D8

2  A9  B9  C9  D9

3 A10 B10 C10 D10

4 A11 B11 C11 D11

>>>
```

hi ghép df1, df2, df3 với axis = 0:

```
>>> pd.concat([df1, df2, df3])
    A   B   C   D
0  A0  B0  C0  D0
1  A1  B1  C1  D1
2  A2  B2  C2  D2
3  A3  B3  C3  D3
0  A4  B4  C4  D4
1  A5  B5  C5  D5
2  A6  B6  C6  D6
3  A7  B7  C7  D7
0  A8  B8  C8  D8
2  A9  B9  C9  D9
3 A10 B10 C10 D10
4 A11 B11 C11 D11
>>>
```

Khi ghép df1,df2 với axis = 1.

```
>>> pd.concat([df1, df2, df3],axis=1)

   A   B   C   D   A   B   C   D   A   B   C   D

0  A0  B0  C0  D0  A4  B4  C4  D4  A8  B8  C8  D8
```

```
1  A1  B1  C1  D1  A5  B5  C5  D5  NaN  NaN  NaN  NaN
2  A2  B2  C2  D2  A6  B6  C6  D6  A9  B9  C9  D9
3  A3  B3  C3  D3  A7  B7  C7  D7  A10  B10  C10  D10
4  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  A11  B11  C11  D11
>>>
```

Từ khóa ignore_index

Ở ví dụ này tôi sử dụng từ khóa **ignore_index=True** để đảm bảo sau khi ghép nối, các index của bảng mới được reset để loại bỏ repeated index labels. Nó tương đương với .reset_index(drop=True). Các bạn hãy thử với các cases sau để so sánh:

(1) df4 = pd.DataFrame({'B': ['B2', 'B3', 'B6', 'B7'],'D': ['D2', 'D3', 'D6', 'D7'],'F': ['F2', 'F3', 'F6', 'F7']},index=[0, 2, 3, 4])

(2) pd.concat([df1, df2]).

(3) pd.concat([df1, df2], ignore_index=True).

(4) pd.concat([df1, df2]).reset_index(drop=True).

và tương tự các cases cho append().

Cả hai concat() và append() cho kết quả giống nhau. NaN sẽ tự điền trong trường hợp thiếu thông tin.

```
>>> pd.concat([df1, df4], ignore_index=True)

   A  B   C  D   F

0  A0  B0  C0  D0  NaN

1  A1  B1  C1  D1  NaN

2  A2  B2  C2  D2  NaN

3  A3  B3  C3  D3  NaN

4  NaN  B2  NaN  D2  F2

5  NaN  B3  NaN  D3  F3

6  NaN  B6  NaN  D6  F6

7  NaN  B7  NaN  D7  F7

>>> df1.append([df4], ignore_index = True)

   A  B   C  D   F

0  A0  B0  C0  D0  NaN

1  A1  B1  C1  D1  NaN

2  A2  B2  C2  D2  NaN

3  A3  B3  C3  D3  NaN

4  NaN  B2  NaN  D2  F2

5  NaN  B3  NaN  D3  F3

6  NaN  B6  NaN  D6  F6

7  NaN  B7  NaN  D7  F7
```

```
>>>
```

Từ khóa keys

Nếu như ta không muốn reset indexes của các dataframes mà muốn đánh dấu index đó đến từ dataframe nào thì giải pháp ở đây chính là sử dụng multi-level indexing qua từ khóa keys.

```
>>> pd.concat([df1,df2],keys=['df1','df2'])

     A  B  C  D

df1 0  A0  B0  C0  D0

   1  A1  B1  C1  D1

   2  A2  B2  C2  D2

   3  A3  B3  C3  D3

df2 0  A4  B4  C4  D4

   1  A5  B5  C5  D5

   2  A6  B6  C6  D6

   3  A7  B7  C7  D7

>>> pd.concat([df1, df2],keys=['df1','df2'],axis=1)

 df1          df2

   A  B  C  D  A  B  C  D

0  A0  B0  C0  D0  A4  B4  C4  D4

1  A1  B1  C1  D1  A5  B5  C5  D5

2  A2  B2  C2  D2  A6  B6  C6  D6

3  A3  B3  C3  D3  A7  B7  C7  D7

>>>
```

Từ khóa join

Ý nghĩa là kết hợp các hàng của nhiều dataframe với nhau. Có nhiều loại join nhưng trong bài này hãy để ý đến hai loại ”inner” và “outer”.

+ outer: sẽ gồm toàn bộ các index của tất cả các dataframe và giá trị NaN sẽ được điền khi thiếu thông tin.

+ inner: sẽ chỉ gồm các index chung của các dataframe.

Tạo thêm df4 để miêu tả rõ ràng.

```
>>> df4 = pd.DataFrame({'B': ['B2', 'B3', 'B6', 'B7'],'D': ['D2', 'D3', 'D6', 'D7'],'F': ['F2', 'F3', 'F6', 'F7']},index=[2, 3, 6, 7])

>>> df1

   A  B  C  D

0  A0  B0  C0  D0

1  A1  B1  C1  D1
```

```
2  A2  B2  C2  D2

3  A3  B3  C3  D3
>>> df4

   B   D   F

2  B2  D2  F2

3  B3  D3  F3

6  B6  D6  F6

7  B7  D7  F7

>>>
```

Từ khóa join_axes

Nếu ta chỉ muốn nhặt một vài index từ các frame để tiến hành ghép nối thì join_axes chính là lựa chọn cho việc này. Ví dụ sau mô tả ta chỉ nhặt các index từ df1 để ghép nối hai frame df1 và df4. NaN sẽ tự điền trong trường hợp thiếu thông tin.

```
>>> pd.concat([df1, df4], axis=1, join_axes=[df1.index])
   A  B  C  D   B   D   F
0  A0  B0  C0  D0  NaN  NaN  NaN
1  A1  B1  C1  D1  NaN  NaN  NaN
2  A2  B2  C2  D2   B2   D2   F2
3  A3  B3  C3  D3   B3   D3   F3
>>>
```

Ngoài cách dùng concat như phía trên, còn một cách khác sử dụng phương thức append() đã rất quen thuộc khi ghép hai list với nhau. Và chúng chỉ có thể ghép các frame theo trục axis = 0.

```
>>> df1.append([df2, df3])
    A   B   C   D
0   A0  B0  C0  D0
1   A1  B1  C1  D1
2   A2  B2  C2  D2
3   A3  B3  C3  D3
4   A4  B4  C4  D4
5   A5  B5  C5  D5
6   A6  B6  C6  D6
7   A7  B7  C7  D7
8   A8  B8  C8  D8
9   A9  B9  C9  D9
10  A10  B10  C10  D10
11  A11  B11  C11  D11
>>>
```

Thực tế ta đã đưa ra các ví dụ về ghép nối các frame có ndim khác nhau. Hãy coi các đối tượng Series như các frame với ndim = 1. Ta có ví dụ ghép nối sau:

```
>>> s1 = pd.Series(['X0', 'X1', 'X2', 'X3'], name='X')
>>> s1.ndim
1
>>> s1

0   X0
```

```
1   X1

2   X2

3   X3

Name: X, dtype: object
>>> df1

   A   B   C   D

0  A0  B0  C0  D0

1  A1  B1  C1  D1

2  A2  B2  C2  D2

3  A3  B3  C3  D3

>>> pd.concat([df1, s1], axis=1)

   A   B   C   D   X

0  A0  B0  C0  D0  X0

1  A1  B1  C1  D1  X1

2  A2  B2  C2  D2  X2

3  A3  B3  C3  D3  X3

>>>
```

Khi dùng append() sẽ được hiểu là thêm hàng vào frame.

```
>>> s2 = pd.Series(['X0', 'X1', 'X2', 'X3'], index=['A', 'B', 'C', 'D'])
>>> s2

A   X0

B   X1

C   X2

D   X3

dtype: object

>>> df1

   A   B   C   D

0  A0  B0  C0  D0

1  A1  B1  C1  D1
```

```
2  A2  B2  C2  D2

3  A3  B3  C3  D3

>>> df1.append(s2, ignore_index=True)

   A   B   C   D

0  A0  B0  C0  D0

1  A1  B1  C1  D1

2  A2  B2  C2  D2

3  A3  B3  C3  D3

4  X0  X1  X2  X3

>>>
```

Ngoài dataframe hoặc series thì append() hoặc pd.concat() cũng chấp nhận đầu vào là dictionary.

Ví dụ ghép một dictionary vào một dataframe qua append(). NaN sẽ tự điền trong trường hợp thiếu thông tin.

```
>>> dicts = [{'A': 1, 'B': 2, 'C': 3, 'X': 4},{'A': 5, 'B': 6, 'C': 7, 'Y': 8}]
>>> df1.append(dicts, ignore_index=True)
   A  B  C   D   X   Y
0  A0  B0  C0   D0  NaN  NaN
1  A1  B1  C1   D1  NaN  NaN
2  A2  B2  C2   D2  NaN  NaN
3  A3  B3  C3   D3  NaN  NaN
4  1   2   3  NaN  4.0  NaN
5  5   6   7  NaN  NaN  8.0
>>>
```

Đến hiện tại, bạn đã nghĩ ra cách tổng hợp dữ liệu từ nhiều files đơn lẻ chưa? Tôi nghĩ là bạn đã trả lời được câu hỏi này, đó là kết hợp các tools mà pandas cung cấp và dùng một loop/comprehension để tổng hợp thành list các dataframe và sử dụng concat() hoặc append() để ghép nối chúng.

Bài toán: Một thư mục chứa 100 files “sales_[0-100].csv” có định dạng *csv cách thức tổng hợp sẽ là:

1. Dùng glob() để liệt kê toàn bộ tên các files “sales_[0-100].csv”.

```
from glob import glob
filenames = glob('sales*.csv')
```

2. Dùng loop hoặc comprehension và read_csv() để đọc toàn bộ các csv files và tạo ra một list các dataframes.

```
dataframes = [pd.read_csv(f) for f in filenames]
```

3. Concat() chúng lại thành một dataframe duy nhất.

```
pd.concat(dataframes)
```

Kết luân

Qua bài học này, bạn đã học được cách kết hợp nhiều dataframe đến từ nhiều nguồn khác nhau trong thế giới thực qua các phương thức **append()** và **pd.concat()** để tạo thành một một dataframe duy nhất giúp cho việc khai phá dữ liệu trở lên dễ dàng hơn. Và chú ý đến ý nghĩa một số đối số quan trọng như **key, axis, join_axes, ignore_index, join** để hiểu cho đúng.