11. Xử lý NaN –missing data

Một vấn đề khi phân tích dữ liệu là xử lý missing data. Pandas đã làm cho việc này dễ dàng nhất có thể.

Series

Tạo một pandas series chứa các giá trị NaN.

Nếu chúng ta nhìn lại một lần nữa trong ví dụ trước, ta có thể thấy rằng chỉ mục của chuỗi giống như các khóa của dictionary mà chúng ta sử dụng để tạo ra các sales. Bây giờ, chúng ta muốn sử dụng một chỉ mục không chồng chéo với các khóa dictionary. Chúng ta đã thấy rằng chúng ta có thể truyền vào list hoặc tuple với đối số từ khoá 'index' để xác định chỉ mục. Trong ví dụ tiếp theo, list (hoặc tuple) được truyền cho tham số keyword 'index' sẽ không tương đương các khoa trong dictionary. Điều này có nghĩa là một số hãng điện thoại từ dictionay sẽ bị thiếu, ví dụ hai hãng ("LG" và "HTC") không xuất hiện trong dictionary.

```
>>> import numpy as np
>>> import pandas as pd
>>> phones = ['Iphone',"Samsung Note","Samsung S","Nokia"]
>>> quantities = [10,12,30,100]
>>> sales = dict(zip(phones, quantities))
>>> S = pd.Series(sales)
>>> brands = ["Iphone","Samsung Note","Nokia","Samsung S","LG","HTC"]
>>> SS = pd.Series(sales,index=brands)
>>> SS
Iphone          10.0
Samsung Note    12.0
Nokia          100.0
Samsung S       30.0
LG               NaN
HTC              NaN
dtype: float64
>>>
```

Chúng ta có thể thấy rằng các brands, mà không có trong từ điển sales, nhận giá trị NaN. NaN là viết tắt của "not a number". Nó cũng có thể được xem là có nghĩa là "missing" trong ví dụ của chúng ta.

Nếu trong quá trình tạo Series mà ta truyền một dictionary, ứng với key có giá trị là “None” thì đối tượng Series tạo thành sẽ nhận giá trị NaN.

```
>>> d = {"a":23, "b":45, "c":None, "d":0}
```

```
>>> S = pd.Series(d)
>>> print(S)
a    23.0
b    45.0
c     NaN
d     0.0
dtype: float64
>>>
```

Dataframe

Tạo một dataframe tồn tại giá trị NaN.

```
>>> import numpy as np
>>> import pandas as pd
>>> df = pd.DataFrame(np.random.randn(5, 4)*100,columns = list("ABCD"))
>>> df
           A          B           C          D
0  193.958156  76.560508 -144.899007  -29.874618
1  196.163029  22.368007   15.188653 -102.040442
2  -22.597828  27.841637  -23.202380   50.040193
3 -111.654480 -25.036501   41.283627  -25.941029
4  -92.212283  77.257557 -122.999865   59.863213
>>> df = df[df>0]
>>> df
           A          B          C          D
0  193.958156  76.560508        NaN        NaN
1  196.163029  22.368007  15.188653        NaN
2         NaN  27.841637        NaN  50.040193
3         NaN        NaN  41.283627        NaN
4         NaN  77.257557        NaN  59.863213
>>>
```

Phương thức isnull() và notnull() để kiểm tra phần tử là NaN hay không là NaN tương ứng.

```
>>> SS.isnull()
```

```
Iphone          False
Samsung Note    False
Nokia           False
Samsung S       False
LG               True
HTC              True
dtype: bool
>>> SS.notnull()
Iphone           True
Samsung Note     True
Nokia            True
Samsung S        True
LG              False
HTC             False
dtype: bool
>>> print "working with dataframe"
working with dataframe
>>> df['A'].isnull()
0    False
1    False
2     True
3     True
4     True
Name: A, dtype: bool
>>> df['A'].notnull()
0     True
1     True
2    False
3    False
4    False
Name: A, dtype: bool
>>> df.isnull()
       A      B      C      D
0  False  False  True  True
```

```
1  False  False  False  True
2  True   False   True  False
3  True   True  False  True
4  True  False   True  False
>>> print “Hang co chua phan tu null”
>>> df.isnull().any()
A    True
B     True
C     True
D    True
dtype: bool
>>> print “Cot co chua phan tu null”
>>> df.isnull().any(1)
0    True
1    True
2     True
3     True
4     True
dtype: bool
```

Loại bỏ missing data dùng **.dropna(axis=0, how=’any’, thresh=None, subset=None, inplace=False)**

Trong đó:

axis: nhận giá trị mặc định là 0 –‘index’, 1-‘columns’

how: ‘any’ sẽ drop axis đang xét nếu có bất gì phần tử NaN nào tồn tại, ‘all’ sẽ drop axis nếu toàn bộ phần tử đang xét nhận giá trị NaN.

```
>>> print “Drop các hàng có chứa phần tử là NaN trong series SS”
Drop các hàng có chứa phần tử là NaN trong series SS
>>> SS.dropna()
Iphone          10.0
Samsung Note    12.0
Nokia          100.0
Samsung S       30.0
dtype: float64
```

```
>>> print "working with dataframe"
working with dataframe
>>> df

          A         B         C         D

0  193.958156  76.560508       NaN       NaN

1  196.163029  22.368007  15.188653       NaN

2         NaN  27.841637       NaN  50.040193

3         NaN       NaN  41.283627       NaN

4         NaN  77.257557       NaN  59.863213

>>> df.dropna()

Empty DataFrame

Columns: [A, B, C, D]

Index: []

>>>
```

Gán gián trị mặc định cho “missing data” dùng hàm fillna(number)

```
>>> SS.fillna(10000000)
Iphone               10.0
Samsung Note         12.0
Nokia               100.0
Samsung S            30.0
LG             10000000.0
HTC            10000000.0
dtype: float64
>>> print "gan qua dictionary"
gan qua dictionary
>>> nafills = {"LG":9919191,"HTC":20}
>>> SS.fillna(nafills)
Iphone               10.0
Samsung Note         12.0
Nokia               100.0
Samsung S            30.0
LG             9919191.0
HTC                  20.0
dtype: float64
>>> print "working with dataframe"
working with dataframe
>>> print ("NaN replaced with '0':")
NaN replaced with '0':
>>> df.fillna(0)
          A         B         C         D
0  193.958156  76.560508   0.000000   0.000000
1  196.163029  22.368007  15.188653   0.000000
2    0.000000  27.841637   0.000000  50.040193
3    0.000000   0.000000  41.283627   0.000000
4    0.000000  77.257557   0.000000  59.863213
>>>
```

Ngoài cách fill các vị trí NaN với một hằng số như phía trên. Pandas còn cung cấp cho ta các lựa chọn khác, fill theo 2 cách sau: sử dụng phương thức pad/ffill để fill dữ liệu kiểu forward tức là missing data sẽ được fill bằng dữ liệu của hàng trước đó, hoặc kiểu thứ hai là bfill/backfill sẽ mang ý nghĩa ngược lại. Cùng xem ví dụ sau:

```
>>> df
          A         B         C         D
0  193.958156  76.560508       NaN       NaN
1  196.163029  22.368007  15.188653       NaN
2         NaN  27.841637       NaN  50.040193
3         NaN       NaN  41.283627       NaN
4         NaN  77.257557       NaN  59.863213


>>> df.fillna(method='bfill')
          A         B         C         D
0  193.958156  76.560508  15.188653  50.040193
1  196.163029  22.368007  15.188653  50.040193
2         NaN  27.841637  41.283627  50.040193
3         NaN  77.257557  41.283627  59.863213
4         NaN  77.257557       NaN  59.863213
>>> df.fillna(method='ffill')
          A         B         C         D
0  193.958156  76.560508       NaN       NaN
1  196.163029  22.368007  15.188653       NaN
2  196.163029  27.841637  15.188653  50.040193
3  196.163029  27.841637  41.283627  50.040193
4  196.163029  77.257557  41.283627  59.863213
>>>
```

Kết Luận

Trong quá trình thao tác trên data, việc tập data đầu vào chưa chuẩn hóa có chứa nhiều vị trí missing là không thể tránh khỏi. Các bạn có thể sử dụng hai phương thức isnull()/notnull() để kiểm tra một phần tử là NaN hay không tương ứng. Sau đó tùy vào bài toán cụ thể mà sử dụng dropna() để drop các phần tử missing hay fillna() để gán giá trị mặc định cho vị trí missing. Chi tiết hơn về dropna() và fillna() các bạn có thể tham khảo thêm các trang sau:

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.dropna.html,

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.fillna.html