When and Why Is Data Missed?

Let us consider an online survey for a product. Many a times, people do not share all the information related to them. Few people share their experience, but not how long they are using the product; few people share how long they are using the product, their experience but not their contact information. Thus, in some or the other way a part of data is always missing, and this is very common in real time.

Let us now see how we can handle missing values (say NA or NaN) using Pandas.

# import the pandas library

import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',

'h'],columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

print df

Its **output** is as follows −

one two three

a 0.077988 0.476149 0.965836

b NaN NaN NaN

c -0.390208 -0.551605 -2.301950

d NaN NaN NaN

e -2.000303 -0.788201 1.510072

f -0.930230 -0.670473 1.146615

g NaN NaN NaN

h 0.085100 0.532791 0.887415

Using reindexing, we have created a DataFrame with missing values. In the output, **NaN** means **Not a Number.**

Check for Missing Values

To make detecting missing values easier (and across different array dtypes), Pandas provides the **isnull()** and **notnull()**functions, which are also methods on Series and DataFrame objects −

Example

import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',

'h'],columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

print df['one'].isnull()

Its **output** is as follows −

a False

b True

c False

d True

e False

f False

g True

h False

Name: one, dtype: bool

Cleaning / Filling Missing Data

Pandas provides various methods for cleaning the missing values. The fillna function can “fill in” NA values with non-null data in a couple of ways, which we have illustrated in the following sections.

Replace NaN with a Scalar Value

The following program shows how you can replace "NaN" with "0".

import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randn(3, 3), index=['a', 'c', 'e'],columns=['one',

'two', 'three'])

df = df.reindex(['a', 'b', 'c'])

print df

print ("NaN replaced with '0':")

print df.fillna(0)

Its **output** is as follows −

one two three

a -0.576991 -0.741695 0.553172

b NaN NaN NaN

c 0.744328 -1.735166 1.749580

NaN replaced with '0':

one two three

a -0.576991 -0.741695 0.553172

b 0.000000 0.000000 0.000000

c 0.744328 -1.735166 1.749580

Here, we are filling with value zero; instead we can also fill with any other value.

Fill NA Forward and Backward

Using the concepts of filling discussed in the ReIndexing Chapter we will fill the missing values.

|  |  |
| --- | --- |
| **Method** | **Action** |
| pad/fill | Fill methods Forward |
| bfill/backfill | Fill methods Backward |

Example

import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',

'h'],columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

print df.fillna(method='pad')

Its **output** is as follows −

one two three

a 0.077988 0.476149 0.965836

b 0.077988 0.476149 0.965836

c -0.390208 -0.551605 -2.301950

d -0.390208 -0.551605 -2.301950

e -2.000303 -0.788201 1.510072

f -0.930230 -0.670473 1.146615

g -0.930230 -0.670473 1.146615

h 0.085100 0.532791 0.887415

Drop Missing Values

If you want to simply exclude the missing values, then use the **dropna** function along with the **axis** argument. By default, axis=0, i.e., along row, which means that if any value within a row is NA then the whole row is excluded.

Example

import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',

'h'],columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

print df.dropna()

Its **output** is as follows −

one two three

a 0.077988 0.476149 0.965836

c -0.390208 -0.551605 -2.301950

e -2.000303 -0.788201 1.510072

f -0.930230 -0.670473 1.146615

h 0.085100 0.532791 0.887415

Replace Missing (or) Generic Values

Many times, we have to replace a generic value with some specific value. We can achieve this by applying the replace method.

Replacing NA with a scalar value is equivalent behavior of the **fillna()** function.

Example

import pandas as pd

import numpy as np

df = pd.DataFrame({'one':[10,20,30,40,50,2000],

'two':[1000,0,30,40,50,60]})

print df.replace({1000:10,2000:60})

Its **output** is as follows −

one two

0 10 10

1 20 0

2 30 30

3 40 40

4 50 50

5 60 60