> 教你学会 Pandas 不是我的目的，**教你轻松玩转 Pandas 才是我的目的。**我会通过一系列实例来带入 Pandas 的知识点，让你在学习 Pandas 的路上不再枯燥。
>
> 声明：我所写的**轻松玩转 Pandas 教程都是免费的**，如果对你有帮助，你可以持续关注我。

在 02-Pandas基本功能详解 (02-Pandas基本功能详解.ipynb) 介绍了 Pandas 中常用的一些功能，使得我们对 Pandas 的使用有了基本的了解。这一章节我们来看下如何使用Pandas处理缺失值。

In [1]:
```
# 导入相关库
import numpy as np
import pandas as pd
```

executed in 7ms, finished 14:54:44 2018-06-18

什么是缺失值

在了解缺失值（也叫控制）如何处理之前，首先要知道的就是什么是缺失值？直观上理解，**缺失值表示的是“缺失的数据”**。

可以思考一个问题：是什么原因造成的缺失值呢？其实有很多原因，实际生活中可能由于有的数据不全所以导致数据缺失，也有可能由于误操作导致数据缺失，又或者人为地造成数据缺失。

来看下我们的示例吧。

In [2]:
```
index = pd.Index(data=["Tom", "Bob", "Mary", "James", "Andy", "Alice"], name="name")

data = {
    "age": [18, 30, np.nan, 40, np.nan, 30],
    "city": ["BeiJing", "ShangHai", "GuangZhou", "ShenZhen", np.nan, " "],
    "sex": [None, "male", "female", "male", np.nan, "unknown"],
    "birth": ["2000-02-10", "1988-10-17", None, "1978-08-08", np.nan, "1988-10-17"]
}

user_info = pd.DataFrame(data=data, index=index)

# 将出生日期转为时间戳
user_info["birth"] = pd.to_datetime(user_info.birth)
user_info
```

executed in 97ms, finished 14:54:44 2018-06-18

Out[2]:

name	age	birth	city	sex
Tom	18.0	2000-02-10	BeiJing	None
Bob	30.0	1988-10-17	ShangHai	male
Mary	NaN	NaT	GuangZhou	female
James	40.0	1978-08-08	ShenZhen	male
Andy	NaN	NaT	NaN	NaN
Alice	30.0	1988-10-17		unknown

可以看到，用户 Tom 的性别为 None，用户 Mary 的年龄为 NAN，生日为 NaT。在 Pandas 的眼中，这些都属于缺失值，可以使用 isnull() 或 notnull() 方法来操作。

In [3]:
```
user_info.isnull()
```

executed in 38ms, finished 14:54:44 2018-06-18

Out[3]:

name	age	birth	city	sex
Tom	False	False	False	True
Bob	False	False	False	False
Mary	True	True	False	False
James	False	False	False	False
Andy	True	True	True	True
Alice	False	False	False	False

除了简单的可以识别出哪些是缺失值或非缺失值外，最常用的就是过滤掉一些缺失的行。比如，我想过滤掉用户年龄为空的用户，如何操作呢？

In [4]:
```
user_info[user_info.age.notnull()]
```

executed in 36ms, finished 14:54:44 2018-06-18

Out[4]:

name	age	birth	city	sex
Tom	18.0	2000-02-10	BeiJing	None
Bob	30.0	1988-10-17	ShangHai	male
James	40.0	1978-08-08	ShenZhen	male
Alice	30.0	1988-10-17		unknown

丢弃缺失值

既然有缺失值了，常见的一种处理办法就是丢弃缺失值。使用 dropna 方法可以丢弃缺失值。

In [5]:
```
user_info.age.dropna()
```

executed in 38ms, finished 14:54:44 2018-06-18

Out[5]:
```
name
Tom      18.0
Bob      30.0
James    40.0
Alice    30.0
Name: age, dtype: float64
```

Seriese 使用 dropna 比较简单，对于 DataFrame 来说，可以设置更多的参数。

axis 参数用于控制行或列，跟其他不一样的是，axis=0 （默认）表示操作行，axis=1 表示操作列。

how 参数可选的值为 any（默认） 或者 all。any 表示一行/列有任意元素为空时即丢弃，all 一行/列所有值都为空时才丢弃。

subset 参数表示删除时只考虑的索引或列名。

thresh参数的类型为整数，它的作用是，比如 thresh=3，会在一行/列中至少有 3 个非空值时将其保留。

In [6]:
```
# 一行数据只要有一个字段存在空值即删除
user_info.dropna(axis=0, how="any")
```

executed in 32ms, finished 14:54:44 2018-06-18

Out[6]:

name	age	birth	city	sex
Bob	30.0	1988-10-17	ShangHai	male
James	40.0	1978-08-08	ShenZhen	male
Alice	30.0	1988-10-17		unknown

In [7]:
```
# 一行数据所有字段都为空值才删除
user_info.dropna(axis=0, how="all")
```

executed in 36ms, finished 14:54:44 2018-06-18

Out[7]:

name	age	birth	city	sex
Tom	18.0	2000-02-10	BeiJing	None
Bob	30.0	1988-10-17	ShangHai	male
Mary	NaN	NaT	GuangZhou	female
James	40.0	1978-08-08	ShenZhen	male
Alice	30.0	1988-10-17		unknown

In [8]:
```
# 一行数据中只要 city 或 sex 存在空值即删除
user_info.dropna(axis=0, how="any", subset=["city", "sex"])
```

executed in 38ms, finished 14:54:45 2018-06-18

Out[8]:

name	age	birth	city	sex
Bob	30.0	1988-10-17	ShangHai	male
Mary	NaN	NaT	GuangZhou	female
James	40.0	1978-08-08	ShenZhen	male
Alice	30.0	1988-10-17		unknown

填充缺失值

除了可以丢弃缺失值外，也可以填充缺失值，最常见的是使用 `fillna` 完成填充。

fillna 这名字一看就是用来填充缺失值的。

填充缺失值时，常见的一种方式是使用一个标量来填充。例如，这里我样有缺失的年龄都填充为 0。

In [9]:
```
user_info.age.fillna(0)
```

executed in 33ms, finished 14:54:45 2018-06-18

Out[9]:
```
name
Tom      18.0
Bob      30.0
Mary      0.0
James    40.0
Andy      0.0
Alice    30.0
Name: age, dtype: float64
```

除了可以使用标量来填充之外，还可以使用前一个或后一个有效值来填充。

设置参数 method='pad' 或 method='ffill' 可以使用前一个有效值来填充。

In [10]:
```
user_info.age.fillna(method="ffill")
```

executed in 36ms, finished 14:54:45 2018-06-18

Out[10]:
```
name
Tom      18.0
Bob      30.0
Mary     30.0
James    40.0
Andy     40.0
Alice    30.0
Name: age, dtype: float64
```

设置参数 method='bfill' 或 method='backfill' 可以使用后一个有效值来填充。

In [11]:
```
user_info.age.fillna(method="backfill")
```

executed in 63ms, finished 14:54:45 2018-06-18

Out[11]:
```
name
Tom      18.0
Bob      30.0
Mary     40.0
James    40.0
Andy     30.0
Alice    30.0
Name: age, dtype: float64
```

除了通过 fillna 方法来填充缺失值外，还可以通过 interpolate 方法来填充。默认情况下使用线性差值，可以是设置 method 参数来改变方式。

In [12]:
```
user_info.age.interpolate()
```

executed in 29ms, finished 14:54:45 2018-06-18

Out[12]:
```
name
Tom      18.0
Bob      30.0
Mary     35.0
James    40.0
Andy     35.0
Alice    30.0
Name: age, dtype: float64
```

替换缺失值

大家有没有想过一个问题：到底什么才是缺失值呢？你可能会奇怪说，前面不是已经说过了么，None、np.nan、NaT 这些都是缺失值。但是我也说过了，这些在 Pandas 的眼中是缺失值，有时候在我们人类的眼中，某些异常值我们也会当做缺失值来处理。

例如，在我们的存储的用户信息中，假定我们限定用户都是青年，出现了年龄为 40 的，我们就可以认为这是一个异常值。再比如，我们都知道性别分为男性（male）和女性（female），在记录用户性别的时候，对于未知的用户性别都记为了 “unknown”,很明显，我们也可以认为“unknown”是缺失值。此外，有的时候会出现空白字符串，这些也可以认为是缺失值。

对于上面的这种情况，我们可以使用 replace 方法来替换缺失值。

In [13]:
```
user_info.age.replace(40, np.nan)
```

executed in 32ms, finished 14:54:45 2018-06-18

Out[13]:
```
name
Tom      18.0
Bob      30.0
Mary      NaN
James     NaN
Andy      NaN
Alice    30.0
Name: age, dtype: float64
```

也可以指定一个映射字典。

In [14]:
```
user_info.age.replace({40: np.nan})
```

executed in 34ms, finished 14:54:45 2018-06-18

Out[14]:
```
name
Tom      18.0
Bob      30.0
Mary      NaN
James     NaN
Andy      NaN
Alice    30.0
Name: age, dtype: float64
```

对于 DataFrame，可以指定每列要替换的值。

In [15]:
```
user_info.replace({"age": 40, "birth": pd.Timestamp("1978-08-08")}, np.nan)
```

executed in 55ms, finished 14:54:45 2018-06-18

Out[15]:

name	age	birth	city	sex
Tom	18.0	2000-02-10	BeiJing	None
Bob	30.0	1988-10-17	ShangHai	male
Mary	NaN	NaT	GuangZhou	female
James	NaN	NaT	ShenZhen	male
Andy	NaN	NaT	NaN	NaN
Alice	30.0	1988-10-17		unknown

类似地，我们可以将特定字符串进行替换，如：将 "unknown" 进行替换。

In [16]:
```
user_info.sex.replace("unknown", np.nan)
```

executed in 22ms, finished 14:54:45 2018-06-18

Out[16]:
```
name
Tom        None
Bob        male
Mary     female
James      male
Andy        NaN
Alice       NaN
Name: sex, dtype: object
```

除了可以替换特定的值之外，还可以使用正则表达式来替换，如：将空白字符串替换成空值。

In [17]:
```
user_info.city.replace(r'\s+', np.nan, regex=True)
```

executed in 41ms, finished 14:54:45 2018-06-18

Out[17]:
```
name
Tom        BeiJing
Bob       ShangHai
Mary     GuangZhou
James     ShenZhen
Andy            NaN
Alice           NaN
Name: city, dtype: object
```

使用其他对象填充

除了我们自己手动丢弃、填充已经替换缺失值之外，我们还可以使用其他对象来填充。

例如有两个关于用户年龄的 Series，其中一个有缺失值，另一个没有，我们可以将没有的缺失值的 Series 中的元素传给有缺失值的。

In [18]:
```
age_new = user_info.age.copy()
age_new.fillna(20, inplace=True)
age_new
```

executed in 34ms, finished 14:54:45 2018-06-18

Out[18]:
```
name
Tom      18.0
Bob      30.0
Mary     20.0
James    40.0
Andy     20.0
Alice    30.0
Name: age, dtype: float64
```

In [19]:
```
user_info.age.combine_first(age_new)
```

executed in 34ms, finished 14:54:45 2018-06-18

Out[19]:
```
name
Tom      18.0
Bob      30.0
Mary     20.0
James    40.0
Andy     20.0
Alice    30.0
Name: age, dtype: float64
```

可以看到，用户信息中关于年龄的缺失值都使用 age_new 这个 Series 填充了。

想要学习更多关于人工智能的知识，请关注公众号：**AI派**

这里我将整篇文章的内容整理成了pdf，想要pdf文件的可以在公众号后台回复关键字：**pandas03。**