> 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。

Pandas 提供了各种方法来完成数据的转换操作，常见的转换操作有：拼接、关联。一起来看看吧。

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

executed in 6ms, finished 16:40:37 2018-08-06

拼接

有两个DataFrame，都存储了用户的一些信息，现在要拼接起来，组成一个DataFrame，如何实现呢？

In [2]:
```
data1 = {
    "name": ["Tom", "Bob"],
    "age": [18, 30],
    "city": ["Bei Jing ", "Shang Hai "]
}

df1 = pd.DataFrame(data=data1)
df1
```

executed in 51ms, finished 16:40:37 2018-08-06

Out[2]:

	age	city	name
0	18	Bei Jing	Tom
1	30	Shang Hai	Bob

In [3]:
```
data2 = {
    "name": ["Mary", "James"],
    "age": [35, 18],
    "city": ["Guang Zhou", "Shen Zhen"]
}

df2 = pd.DataFrame(data=data2)
df2
```

executed in 29ms, finished 16:40:37 2018-08-06

Out[3]:

	age	city	name
0	35	Guang Zhou	Mary
1	18	Shen Zhen	James

append

append 是最简单的拼接两个DataFrame的方法。

In [4]:
```
df1.append(df2)
```

executed in 32ms, finished 16:40:37 2018-08-06

Out[4]:

	age	city	name
0	18	Bei Jing	Tom
1	30	Shang Hai	Bob
0	35	Guang Zhou	Mary
1	18	Shen Zhen	James

可以看到，拼接后的索引默认还是原有的索引，如果想要重新生成索引的话，设置参数 ignore_index=True 即可。

In [5]:
```
df1.append(df2, ignore_index=True)
```

executed in 28ms, finished 16:40:37 2018-08-06

Out[5]:

	age	city	name
0	18	Bei Jing	Tom
1	30	Shang Hai	Bob
2	35	Guang Zhou	Mary
3	18	Shen Zhen	James

concat

除了 append 这种方式之外，还有 concat 这种方式可以实现相同的功能。

In [6]:
```
objs=[df1, df2]
pd.concat(objs, ignore_index=True)
```

executed in 36ms, finished 16:40:37 2018-08-06

Out[6]:

	age	city	name
0	18	Bei Jing	Tom
1	30	Shang Hai	Bob
2	35	Guang Zhou	Mary
3	18	Shen Zhen	James

如果想要区分出不同的DataFrame的数据，可以通过设置参数 keys，当然得设置参数 ignore_index=False。

In [7]:
```
pd.concat(objs, ignore_index=False, keys=["df1", "df2"])
```

executed in 48ms, finished 16:40:37 2018-08-06

Out[7]:

		age	city	name
df1	**0**	18	Bei Jing	Tom
	1	30	Shang Hai	Bob
df2	**0**	35	Guang Zhou	Mary
	1	18	Shen Zhen	James

关联

有两个DataFrame，分别存储了用户的部分信息，现在需要将用户的这些信息关联起来，如何实现呢？

In [8]:
```
data1 = {
    "name": ["Tom", "Bob", "Mary", "James"],
    "age": [18, 30, 35, 18],
    "city": ["Bei Jing ", "Shang Hai ", "Guang Zhou", "Shen Zhen"]
}

df1 = pd.DataFrame(data=data1)
df1
```

executed in 40ms, finished 16:40:37 2018-08-06

Out[8]:

	age	city	name
0	18	Bei Jing	Tom
1	30	Shang Hai	Bob
2	35	Guang Zhou	Mary
3	18	Shen Zhen	James

In [9]:
```
data2 = {"name": ["Bob", "Mary", "James", "Andy"],
        "sex": ["male", "female", "male", np.nan],
         "income": [8000, 8000, 4000, 6000]
}

df2 = pd.DataFrame(data=data2)
df2
```

executed in 38ms, finished 16:40:37 2018-08-06

Out[9]:

	income	name	sex
0	8000	Bob	male
1	8000	Mary	female
2	4000	James	male
3	6000	Andy	NaN

merge

通过 pd.merge 可以关联两个DataFrame，这里我们设置参数 on="name"，表示依据 name 来作为关联键。

In [10]:
```
pd.merge(df1, df2, on="name")
```

executed in 45ms, finished 16:40:37 2018-08-06

Out[10]:

	age	city	name	income	sex
0	30	Shang Hai	Bob	8000	male
1	35	Guang Zhou	Mary	8000	female
2	18	Shen Zhen	James	4000	male

关联后发现数据变少了，只有 3 行数据，这是因为默认关联的方式是 inner，如果不想丢失任何数据，可以设置参数 how="outer"。

In [11]:
```
pd.merge(df1, df2, on="name", how="outer")
```

executed in 43ms, finished 16:40:37 2018-08-06

Out[11]:

	age	city	name	income	sex
0	18.0	Bei Jing	Tom	NaN	NaN
1	30.0	Shang Hai	Bob	8000.0	male
2	35.0	Guang Zhou	Mary	8000.0	female
3	18.0	Shen Zhen	James	4000.0	male
4	NaN	NaN	Andy	6000.0	NaN

可以看到，设置参数 how="outer" 后，确实不会丢失任何数据，他会在不存在的地方填为缺失值。

如果我们想保留左边所有的数据，可以设置参数 `how="left"`；反之，如果想保留右边的所有数据，可以设置参数 `how="right"`

In [12]:
```
pd.merge(df1, df2, on="name", how="left")
```

executed in 39ms, finished 16:40:37 2018-08-06

Out[12]:

	age	city	name	income	sex
0	18	Bei Jing	Tom	NaN	NaN
1	30	Shang Hai	Bob	8000.0	male
2	35	Guang Zhou	Mary	8000.0	female
3	18	Shen Zhen	James	4000.0	male

有时候，两个 DataFrame 中需要关联的键的名称不一样，可以通过 `left_on` 和 `right_on` 来分别设置。

In [13]:
```
df1.rename(columns={"name": "name1"}, inplace=True)
df1
```

executed in 31ms, finished 16:40:37 2018-08-06

Out[13]:

	age	city	name1
0	18	Bei Jing	Tom
1	30	Shang Hai	Bob
2	35	Guang Zhou	Mary
3	18	Shen Zhen	James

In [14]:
```
df2.rename(columns={"name": "name2"}, inplace=True)
df2
```

executed in 31ms, finished 16:40:37 2018-08-06

Out[14]:

	income	name2	sex
0	8000	Bob	male
1	8000	Mary	female
2	4000	James	male
3	6000	Andy	NaN

In [15]:
```
pd.merge(df1, df2, left_on="name1", right_on="name2")
```

executed in 37ms, finished 16:40:37 2018-08-06

Out[15]:

	age	city	name1	income	name2	sex
0	30	Shang Hai	Bob	8000	Bob	male
1	35	Guang Zhou	Mary	8000	Mary	female
2	18	Shen Zhen	James	4000	James	male

有时候，两个DataFrame中都包含相同名称的字段，如何处理呢？

我们可以设置参数 `suffixes`，默认 `suffixes=('_x', '_y')` 表示将相同名称的左边的DataFrame的字段名加上后缀 `_x`，右边加上后缀 `_y`。

In [16]:
```
df1["sex"] = "male"
df1
```

executed in 32ms, finished 16:40:37 2018-08-06

Out[16]:

	age	city	name1	sex
0	18	Bei Jing	Tom	male
1	30	Shang Hai	Bob	male
2	35	Guang Zhou	Mary	male
3	18	Shen Zhen	James	male

In [17]:
```
pd.merge(df1, df2, left_on="name1", right_on="name2")
```

executed in 32ms, finished 16:40:37 2018-08-06

Out[17]:

	age	city	name1	sex_x	income	name2	sex_y
0	30	Shang Hai	Bob	male	8000	Bob	male
1	35	Guang Zhou	Mary	male	8000	Mary	female
2	18	Shen Zhen	James	male	4000	James	male

In [18]:
```
pd.merge(df1, df2, left_on="name1", right_on="name2", suffixes=("_left", "_right"))
```

executed in 33ms, finished 16:40:37 2018-08-06

Out[18]:

	age	city	name1	sex_left	income	name2	sex_right
0	30	Shang Hai	Bob	male	8000	Bob	male
1	35	Guang Zhou	Mary	male	8000	Mary	female
2	18	Shen Zhen	James	male	4000	James	male

join

除了 merge 这种方式外，还可以通过 join 这种方式实现关联。相比 merge，join 这种方式有以下几个不同：

- 默认参数on=None，表示关联时使用左边和右边的索引作为键，设置参数on可以指定的是关联时左边的所用到的键名
- 左边和右边字段名称重复时，通过设置参数 lsuffix 和 rsuffix 来解决。

In [19]:
```
df1.join(df2.set_index("name2"), on="name1", lsuffix="_left")
```

executed in 30ms, finished 16:40:37 2018-08-06

Out[19]:

	age	city	name1	sex_left	income	sex
0	18	Bei Jing	Tom	male	NaN	NaN
1	30	Shang Hai	Bob	male	8000.0	male
2	35	Guang Zhou	Mary	male	8000.0	female
3	18	Shen Zhen	James	male	4000.0	male

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：**pandas**。