

Pandas | 详解数据的合并和拼接

原创 石头 机器学习算法那些事 2019-04-28

本文介绍了利用pandas包的merge、join和concat方法来完成数据的合并和拼接，merge方法主要是基于两个dataframe的共同列进行合并，join方法主要是基于两个dataframe的索引进行合并，concat方法是对series或dataframe进行行拼接或列拼接，本文详细分析了上面三种方法的合并和拼接操作。

目录

1. Merge方法
2. Join方法
3. concat方法
4. 小结

1. Merge方法

pandas的merge方法是基于共同列，将两个dataframe连接起来。下面分析merge方法的主要参数含义：

left/right: 左/右位置的dataframe。

how: 数据合并的方式。left: 基于左dataframe列的数据合并；right: 基于右dataframe列的数据合并；outer: 基于列的数据外合并（取并集）；inner: 基于列的数据内合并（取交集）；默认为'inner'。

on: 用来合并的列名，这个参数需要保证两个dataframe有相同的列名。

left_on/right_on: 左/右dataframe合并的列名，也可为索引，数组和列表。

left_index/right_index: 是否以index作为数据合并的列名，True表示是。

sort: 根据dataframe合并的keys排序，默认是。

suffixes: 若有相同列且该列没有作为合并的列，可通过suffixes设置该列的后缀名，一般为元组和列表类型。

merges通过设置how参数选择两个dataframe的连接方式，有内连接，外连接，左连接，右连接，下面通过例子介绍连接的含义。

1.1 内连接

how='inner'，dataframe的链接方式为内连接，我们可以理解基于共同列的交集进行连接，参数on设置连接的共有列名。

```
# 单列的内连接
# 定义df1
df1 = pd.DataFrame({'alpha': ['A', 'B', 'B', 'C', 'D', 'E'], 'feature1': [1, 1, 2, 3, 3, 1],
                    'feature2': ['low', 'medium', 'medium', 'high', 'low', 'high']})
# 定义df2
df2 = pd.DataFrame({'alpha': ['A', 'A', 'B', 'F'], 'pazham': ['apple', 'orange', 'pine', 'pear'],
                    'kilo': ['high', 'low', 'high', 'medium'], 'price': np.array([5, 6, 5, 7])})
```

```
# print(df1)
# print(df2)
# 基于共同列alpha的内连接
df3 = pd.merge(df1, df2, how='inner', on='alpha')
df3

#>
   alpha  feature1  feature2  pazham  kilo  price
0      A         1        low   apple  high     5
1      A         1        low  orange  low     6
2      B         1      medium   pine   high     5
3      B         2      medium   pine   high     5
```

下面图解内连接的含义：

df1				df2				df3								
alpha	feature1	feature2		alpha	pazham	kilo	price		alpha	feature1	feature2	pazham	kilo	price		
0	A	1	low	0	A	apple	high	5	内连接→	0	A	1	low	apple	high	5
1	B	1	medium	1	A	orange	low	6		1	A	1	low	orange	low	6
2	B	2	medium	2	B	pine	high	5		2	B	1	medium	pine	high	5
3	C	3	high	3	F	pear	medium	7		3	B	2	medium	pine	high	5
4	D	3	low													
5	E	1	high													

取共同列alpha值的交集进行连接。

1.2 外连接

how='outer', dataframe的链接方式为外连接，我们可以理解基于共同列的并集进行连接，参数on设置连接的共有列名。

```
# 单列的外连接
# 定义df1
df1 = pd.DataFrame({'alpha': ['A', 'B', 'B', 'C', 'D', 'E'], 'feature1': [1, 1, 2, 3, 3, 1],
                    'feature2': ['low', 'medium', 'medium', 'high', 'low', 'high']})
# 定义df2
df2 = pd.DataFrame({'alpha': ['A', 'A', 'B', 'F'], 'pazham': ['apple', 'orange', 'pine', 'pear'],
                    'kilo': ['high', 'low', 'high', 'medium'], 'price': np.array([5, 6, 5, 7])})
# 基于共同列alpha的内连接
df4 = pd.merge(df1, df2, how='outer', on='alpha')
df4

#>
```

```
   alpha  feature1  feature2  pazham  kilo  price
0      A         1.0      low   apple  high     5.0
1      A         1.0      low  orange  low     6.0
2      B         1.0      medium  pine   high     5.0
3      B         2.0      medium  pine   high     5.0
4      C         3.0      high   NaN   NaN     NaN
5      D         3.0      low   NaN   NaN     NaN
6      E         1.0      high   NaN   NaN     NaN
7      F         NaN     NaN   pear   medium     7.0
```

下面图解外连接的含义：

df1				df2				df5							
alpha	feature1	feature2		alpha	pazham	kilo	price	alpha	feature1	feature2	pazham	kilo	price		
0	A	1	low	0	A	apple	high	5	0	A	1.0	low	apple	high	5.0
1	B	1	medium	1	A	orange	low	6	1	A	1.0	low	orange	low	6.0
2	B	2	medium	2	B	pine	high	5	2	B	1.0	medium	pine	high	5.0
3	C	3	high	3	F	pear	medium	7	3	B	2.0	medium	pine	high	5.0
4	D	3	low						4	C	3.0	high	NaN	NaN	NaN
5	E	1	high						5	D	3.0	low	NaN	NaN	NaN
									6	E	1.0	high	NaN	NaN	NaN
									7	F	NaN	NaN	pear	medium	7.0

若两个dataframe间除了on设置的连接列外并无相同列，则该列的值置为NaN。

1.3 左连接

how='left'，dataframe的链接方式为左连接，我们可以理解基于左边位置dataframe的列进行连接，参数on设置连接的共有列名。

```
# 单列的左连接
# 定义df1
df1 = pd.DataFrame({'alpha': ['A', 'B', 'B', 'C', 'D', 'E'], 'feature1': [1, 1, 2, 3, 3, 1],
                    'feature2': ['low', 'medium', 'medium', 'high', 'low', 'high']})
# 定义df2
df2 = pd.DataFrame({'alpha': ['A', 'A', 'B', 'F'], 'pazham': ['apple', 'orange', 'pine', 'pear'],
                    'kilo': ['high', 'low', 'high', 'medium'], 'price': np.array([5, 6, 5, 7])})
# 基于共同列alpha的左连接
df5 = pd.merge(df1, df2, how='left', on='alpha')
df5
```

```
#>
   alpha  feature1  feature2  pazham  kilo  price
0     A         1      low     apple  high    5.0
1     A         1      low     orange  low    6.0
2     B         1    medium     pine   high    5.0
3     B         2    medium     pine   high    5.0
4     C         3      high      NaN   NaN    NaN
5     D         3      low      NaN   NaN    NaN
6     E         1      high      NaN   NaN    NaN
```

下面图解左连接的含义：

df1				df2				df5							
alpha	feature1	feature2		alpha	pazham	kilo	price	alpha	feature1	feature2	pazham	kilo	price		
0	A	1	low	0	A	apple	high	5	0	A	1	low	apple	high	5.0
1	B	1	medium	1	A	orange	low	6	1	A	1	low	orange	low	6.0
2	B	2	medium	2	B	pine	high	5	2	B	1	medium	pine	high	5.0
3	C	3	high	3	F	pear	medium	7	3	B	2	medium	pine	high	5.0
4	D	3	low						4	C	3	high	NaN	NaN	NaN
5	E	1	high						5	D	3	low	NaN	NaN	NaN
									6	E	1	high	NaN	NaN	NaN

因为df2的连接列alpha有两个'A'值，所以左连接的df5有两个'A'值，若两个dataframe间除了on设置的连接列外并无相同列，则该列的值置为NaN。

1.4 右连接

how='left'，dataframe的链接方式为左连接，我们可以理解基于右边位置dataframe的列进行连接，参数on设置连接的共有列名。

```
# 单列的右连接
# 定义df1
df1 = pd.DataFrame({'alpha':['A','B','B','C','D','E'],'feature1':[1,1,2,3,3,1],
                    'feature2':['low','medium','medium','high','low','high']})
# 定义df2
df2 = pd.DataFrame({'alpha':['A','A','B','F'],'pazham':['apple','orange','pine','pear'],
                    'kilo':['high','low','high','medium'],'price':np.array([5,6,5,7])})
# 基于共同列alpha的右连接
df6 = pd.merge(df1,df2,how='right',on='alpha')
df6

#>
```

	alpha	feature1	feature2	pazham	kilo	price
0	A	1.0	low	apple	high	5
1	A	1.0	low	orange	low	6
2	B	1.0	medium	pine	high	5
3	B	2.0	medium	pine	high	5
4	F	NaN	NaN	pear	medium	7

下面图解左连接的含义：



因为df1的连接列alpha有两个'B'值，所以右连接的df6有两个'B'值。若两个dataframe间除了on设置的连接列外并无相同列，则该列的值置为NaN。

1.5 基于多列的连接算法

多列连接的算法与单列连接一致，本节只介绍基于多列的内连接和右连接，读者可自己编码并按照本文给出的图解方式去理解外连接和左连接。

多列的内连接：

```
# 多列的内连接
# 定义df1
```

```
df1 = pd.DataFrame({'alpha':['A','B','B','C','D','E'], 'beta':['a','a','b','c','c','e'],
                    'feature1':[1,1,2,3,3,1], 'feature2':['low','medium','medium','high','low','low'],
                    'pazham':['apple','apple','orange','apple','apple','apple'],
                    'kilo':['high','low','high','medium','high','high'], 'price':np.array([5,6,5,7])})

# 定义df2
df2 = pd.DataFrame({'alpha':['A','A','B','F'], 'beta':['d','d','b','f'], 'pazham':['apple','orange','apple','apple'],
                    'kilo':['high','low','high','medium'], 'price':np.array([5,6,5,7])})

# 基于共同列alpha和beta的内连接
df7 = pd.merge(df1, df2, on=['alpha', 'beta'], how='inner')
df7
```

```
#>
   alpha beta feature1 feature2 pazham kilo price
0      B   b         2    medium    pine  high    5
```

图解多列内连接的方法：

df1					df2					df 7						
alpha	beta	feature1	feature2		alpha	beta	pazham	kilo	price	alpha	beta	feature1	feature2	pazham	kilo	price
0	A	a	1	low	0	A	d	apple	high	5	0	B	b	2	medium	pine high 5
1	B	a	1	medium	1	A	d	orange	low	6						
2	B	b	2	medium	2	B	b	pine	high	5						
3	C	c	3	high	3	F	f	pear	medium	7						
4	D	c	3	low												
5	E	e	1	high												

多列的右连接：

```
# 多列的右连接
# 定义df1
df1 = pd.DataFrame({'alpha':['A','B','B','C','D','E'], 'beta':['a','a','b','c','c','e'],
                    'feature1':[1,1,2,3,3,1], 'feature2':['low','medium','medium','high','low','low'],
                    'pazham':['apple','apple','orange','apple','apple','apple'],
                    'kilo':['high','low','high','medium','high','high'], 'price':np.array([5,6,5,7])})

# 定义df2
df2 = pd.DataFrame({'alpha':['A','A','B','F'], 'beta':['d','d','b','f'], 'pazham':['apple','orange','apple','apple'],
                    'kilo':['high','low','high','medium'], 'price':np.array([5,6,5,7])})

# 基于共同列alpha和beta的右连接
df8 = pd.merge(df1, df2, on=['alpha', 'beta'], how='right')
df8
```

```
#>
   alpha beta feature1 feature2 pazham kilo price
0      B   b         2.0    medium    pine  high    5
1      A   d         NaN         NaN    apple  high    5
2      A   d         NaN         NaN    orange  low    6
3      F   f         NaN         NaN    pear   medium  7
```

图解多列的右连接方法：

df1					df2					右连接	df 8							
alpha	beta	feature1	feature2		alpha	beta	pazham	kilo	price	→	alpha	beta	feature1	feature2	pazham	kilo	price	
0	A	a	1	low	0	A	d	apple	high	5	0	B	b	2.0	medium	pine	high	5
1	B	a	1	medium	1	A	d	orange	low	6	1	A	d	NaN	NaN	apple	high	5
2	B	b	2	medium	2	B	b	pine	high	5	2	A	d	NaN	NaN	orange	low	6
3	C	c	3	high	3	F	f	pear	medium	7	3	F	f	NaN	NaN	pear	medium	7
4	D	c	3	low														
5	E	e	1	high														

1.6 基于index的连接方法

前面介绍了基于column的连接方法，merge方法亦可基于index连接dataframe。

```
# 基于column和index的右连接
# 定义df1
df1 = pd.DataFrame({'alpha':['A','B','B','C','D','E'],'beta':['a','a','b','c','c','e'],
                    'feature1':[1,1,2,3,3,1],'feature2':['low','medium','medium','high','low','l']})

# 定义df2
df2 = pd.DataFrame({'alpha':['A','A','B','F'],'pazham':['apple','orange','pine','pear'],
                    'kilo':['high','low','high','medium'],'price':np.array([5,6,5,7])}, index=[0,1,2,3])

# 基于df1的beta列和df2的index连接
df9 = pd.merge(df1, df2, how='inner', left_on='beta', right_index=True)
df9
```

```
#>
   alpha_x  beta  feature1  feature2  alpha_y  pazham  kilo  price
2      B    b         2    medium      B    pine   high     5
```

图解index和column的内连接方法：

df1					df2					df9								
alpha	beta	feature1	feature2		alpha	pazham	kilo	price	内连接	alpha_x	beta	feature1	feature2	alpha_y	pazham	kilo	price	
0	A	a	1	low	d	A	apple	high	5	2	B	b	2	medium	B	pine	high	5
1	B	a	1	medium	d	A	orange	low	6									
2	B	b	2	medium	b	B	pine	high	5									
3	C	c	3	high	f	F	pear	medium	7									
4	D	c	3	low														
5	E	e	1	high														

设置参数suffixes以修改除连接列外相同列的后缀名。

```
# 基于df1的alpha列和df2的key内连接
df9 = pd.merge(df1, df2, how='inner', left_on='beta', right_index=True, suffixes=('_df1', '_df2'))
df9
```

```
#>
   alpha_df1  beta  feature1  feature2  alpha_df2  pazham  kilo  price
2      B    b         2    medium      B    pine   high     5
```

2. join方法

join方法是基于index连接dataframe，merge方法是基于column连接，连接方法有内连接，外连接，左连接和右连接，与merge一致。

index与index的连接：

```
caller = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3', 'K4', 'K5'], 'A': ['A0', 'A1', 'A2', 'A3', 'A4', 'A5']})
other = pd.DataFrame({'key': ['K0', 'K1', 'K2'], 'B': ['B0', 'B1', 'B2']})
# lsuffix和rsuffix设置连接的后缀名
caller.join(other, lsuffix='_caller', rsuffix='_other', how='inner')
```

```
#>
```

	key_caller	A	key_other	B
0	K0	A0	K0	B0
1	K1	A1	K1	B1
2	K2	A2	K2	B2

join也可以基于列进行连接：

```
caller = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3', 'K4', 'K5'], 'A': ['A0', 'A1', 'A2', 'A3', 'A4', 'A5']})
other = pd.DataFrame({'key': ['K0', 'K1', 'K2'], 'B': ['B0', 'B1', 'B2']})
# 基于key列进行连接
caller.set_index('key').join(other.set_index('key'), how='inner')
```

```
#>
```

	A	B
key		
K0	A0	B0
K1	A1	B1
K2	A2	B2

因此，join和merge的连接方法类似，这里就不展开join方法了，建议大家使用merge方法。

3. concat方法

concat方法是拼接函数，有行拼接和列拼接，默认是行拼接，拼接方法默认是外拼接（并集），拼接的对象是pandas数据类型。

3.1 series类型的拼接方法

行拼接：

```
df1 = pd.Series([1.1, 2.2, 3.3], index=['i1', 'i2', 'i3'])
df2 = pd.Series([4.4, 5.5, 6.6], index=['i2', 'i3', 'i4'])

# 行拼接
pd.concat([df1, df2])
```

```
#>
```

	i1	i2	i3	i4
1.1	1.1	2.2	3.3	
2.2		4.4	5.5	6.6

```
i3    3.3
i2    4.4
i3    5.5
i4    6.6
dtype: float64
```

行拼接若有相同的索引，为了区分索引，我们在最外层定义了索引的分组情况。

```
# 对行拼接分组
pd.concat([df1, df2], keys=['fea1', 'fea2'])
```

```
fea1  i1    1.1
      i2    2.2
      i3    3.3
fea2  i2    4.4
      i3    5.5
      i4    6.6
dtype: float64
```

列拼接：

默认以并集的方式拼接

```
# 列拼接，默认是并集
pd.concat([df1, df2], axis=1)

#>
      0      1
i1    1.1  NaN
i2    2.2  4.4
i3    3.3  5.5
i4    NaN  6.6
```

以交集的方式拼接：

```
# 列拼接的内连接（交）
pd.concat([df1, df2], axis=1, join='inner')

#>
      0      1
i2    2.2  4.4
i3    3.3  5.5
```

设置列拼接的列名：

```
# 列拼接的内连接（交）
pd.concat([df1, df2], axis=1, join='inner', keys=['fea1', 'fea2'])

#>
      fea1  fea2
i2    2.2    4.4
i3    3.3    5.5
```

对指定的索引拼接：


```
# 指定索引[i1, i2, i3]的列拼接
pd.concat([df1, df2], axis=1, join_axes=[['i1', 'i2', 'i3']])

#>
      0      1
i1  1. 1  NaN
i2  2. 2   4. 4
i3  3. 3   5. 5
```

3.2 dataframe类型的拼接方法

行拼接：

```
df1 = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3', 'K4', 'K5'], 'A': ['A0', 'A1', 'A2', 'A3',
df2 = pd.DataFrame({'key': ['K0', 'K1', 'K2'], 'B': ['B0', 'B1', 'B2']})
# 行拼接
pd.concat([df1, df2])

#>
      A      B      key
0  A0  NaN  K0
1  A1  NaN  K1
2  A2  NaN  K2
3  A3  NaN  K3
4  A4  NaN  K4
5  A5  NaN  K5
0  NaN  B0  K0
1  NaN  B1  K1
2  NaN  B2  K2
```

列拼接：

```
# 列拼接
pd.concat([df1, df2], axis=1)

#>
      key  A      key  B
0  K0  A0  K0  B0
1  K1  A1  K1  B1
2  K2  A2  K2  B2
3  K3  A3  NaN  NaN
4  K4  A4  NaN  NaN
5  K5  A5  NaN  NaN
```

若列拼接或行拼接有重复的列名和行名，则报错：

```
# 判断是否有重复的列名，若有则报错
pd.concat([df1, df2], axis=1, verify_integrity = True)

#>
ValueError: Indexes have overlapping values: Index(['key'], dtype='object')
```

4. 小结

本文介绍的合并和拼接方法都有共通的联系点，merge和join方法基本上能实现相同的功能，我建议用merge完成相关的合并功能，本文的源码已上传github，附链接。

<https://github.com/zhangleiszu/pandas>

参考

https://segmentfault.com/a/1190000018537597?utm_source=tag-newest

<https://www.cnblogs.com/bigshow1949/p/7016235.html>

推荐阅读

精心整理 | 非常全面的Pandas入门教程

