Lesson Pandas-Merge-Join-Concat

November 16, 2021

Table of Contents

1 Merge

1.1 Making the 'table' wider (i.e., adding columns from. a second source)

1.1.1 Basic merge - uses the index

1.1.2 Specifying columns to use for the merge

1.1.3 Column names not the same

1.1.4 Outer, Left & Right join

2 Same code using .join

2.1 Append (aka union)

2.2 Concatenate

2.2.0.1 Concat can work along either axis

[19]:
```
import pandas as pd
```

1 Merge

1.1 Making the 'table' wider (i.e., adding columns from. a second source)

- merge() for combining data on common columns or indices
- .join() for combining data on a key column or an index (uses merge internally, faster because index is used)
- concat() for combining DataFrames across rows or columns

More info: https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html#brief-primer-on-merge-methods-relational-algebra

[20]:
```
df1 = {
    'location':['bolton','berlin','boyleston','charlton'],
    'apples': [3, 2, 0, 1],
    'pears': [0, 3, 7, 2]
}

df2 = {
    'location':['bolton','berlin','boyleston','charlton'],
```

```
    'blueberries': [3, 2, 0, 1],
    'strawberries': [0, 3, 7, 2]
}

d1 = pd.DataFrame(df1)
d2 = pd.DataFrame(df2)
print(d1)
print(' ')
print(d2)
```

```
    location  apples  pears
0     bolton       3      0
1     berlin       2      3
2  boyleston       0      7
3   charlton       1      2

    location  blueberries  strawberries
0     bolton            3             0
1     berlin            2             3
2  boyleston            0             7
3   charlton            1             2
```

1.1.1 Basic merge - uses the index

[21]:
```
pd.merge(d1,d2,how = 'inner')  # Inner, outer, left, right
# When not being explicit, the merge is based on the index for each dataframe.
```

[21]:
```
    location  apples  pears  blueberries  strawberries
0     bolton       3      0            3             0
1     berlin       2      3            2             3
2  boyleston       0      7            0             7
3   charlton       1      2            1             2
```

1.1.2 Specifying columns to use for the merge

[22]:
```
df3 = {
    'state':['MA','MA','VT','VT'],
    'location':['bolton','berlin','boyleston','berlin'],
    'apples': [3, 2, 0, 1],
    'pears': [0, 3, 7, 2]
}

df4 = {
    'state':['MA','MA','VT','VT'],
    'location':['bolton','berlin','boyleston','berlin'],
    'blueberries': [3, 2, 0, 1],
    'strawberries': [0, 3, 7, 2]
}
```

```
d3 = pd.DataFrame(df3)
d4 = pd.DataFrame(df4)
```

[23]:
```
multi = pd.merge(d3,d4, how = 'inner', on = ['state','location'])
multi
```

[23]:

	state	location	apples	pears	blueberries	strawberries
0	MA	bolton	3	0	3	0
1	MA	berlin	2	3	2	3
2	VT	boyleston	0	7	0	7
3	VT	berlin	1	2	1	2

1.1.3 Column names not the same

[24]:
```
df5 = {
    'state':['MA','MA','VT','VT'],
    'location':['bolton','berlin','boyleston','berlin'],
    'apples': [3, 2, 0, 1],
    'pears': [0, 3, 7, 2]
}

df6 = {
    'states':['MA','MA','VT','VT'],
    'loc':['bolton','berlin','boyleston','berlin'],
    'blueberries': [3, 2, 0, 1],
    'strawberries': [0, 3, 7, 2]
}

d5 = pd.DataFrame(df5)
d6 = pd.DataFrame(df6)
```

[25]:
```
# Use left_on, right_on instead of on.

pd.merge(d5, d6, left_on = ['state','location'], right_on = ['states','loc'])

# Would it be better or easier to just rename the columns?
```

[25]:

	state	location	apples	pears	states	loc	blueberries	strawberries
0	MA	bolton	3	0	MA	bolton	3	0
1	MA	berlin	2	3	MA	berlin	2	3
2	VT	boyleston	0	7	VT	boyleston	0	7
3	VT	berlin	1	2	VT	berlin	1	2

1.1.4 Outer, Left & Right join

[26]:
```
# Data for d7 and d8 does not 'line up' cleanly

df7 = {
    'state':['MA','MA','VT','NH'],
    'location':['bolton','berlin','boyleston','berlin'],
    'apples': [3, 2, 0, 1],
    'pears': [0, 3, 7, 2]
}

df8 = {
    'state':['MA','MA','VT','ME'],
    'location':['bolton','berlin','boyleston','berlin'],
    'blueberries': [3, 2, 0, 1],
    'strawberries': [0, 3, 7, 2]
}

d7 = pd.DataFrame(df7)
d8 = pd.DataFrame(df8)
```

[27]:
```
# Outer join
outer = pd.merge(d7,d8, how = 'outer', on = ['state','location'])
outer
```

[27]:

	state	location	apples	pears	blueberries	strawberries
0	MA	bolton	3.0	0.0	3.0	0.0
1	MA	berlin	2.0	3.0	2.0	3.0
2	VT	boyleston	0.0	7.0	0.0	7.0
3	NH	berlin	1.0	2.0	NaN	NaN
4	ME	berlin	NaN	NaN	1.0	2.0

[28]:
```
# Left join
left = pd.merge(d7,d8, how = 'left', on = ['state','location'])
left

# Notice the Nan values
```

[28]:

	state	location	apples	pears	blueberries	strawberries
0	MA	bolton	3	0	3.0	0.0
1	MA	berlin	2	3	2.0	3.0
2	VT	boyleston	0	7	0.0	7.0
3	NH	berlin	1	2	NaN	NaN

[29]:
```
# Right join
right = pd.merge(d7,d8, how = 'right', on = ['state','location'])
right
```

[29]:

	state	location	apples	pears	blueberries	strawberries
0	MA	bolton	3.0	0.0	3	0
1	MA	berlin	2.0	3.0	2	3
2	VT	boyleston	0.0	7.0	0	7
3	ME	berlin	NaN	NaN	1	2

2 Same code using .join

[30]:
```
# basic syntax - first_dataframe.join(to second dataframe)
d1.join(d2, lsuffix = '_1')

# d1.join(d2, lsuffix = '_1', rsuffix = '_2')


# If you want to use join() and want to merge the columns, you must set them to␣
 ↪be indexes first.

d1.join(d2.set_index('location'), on='location', lsuffix = '_1', rsuffix = '_2')


d3.join(d4.set_index(['state','location']), on=['state','location'],␣
 ↪lsuffix='_3')
```

[30]:

	state	location	apples	pears	blueberries	strawberries
0	MA	bolton	3	0	3	0
1	MA	berlin	2	3	2	3
2	VT	boyleston	0	7	0	7
3	VT	berlin	1	2	1	2

[31]:
```
# Outer join

d7.join(d8.set_index(['state','location']), on=['state','location'],␣
 ↪lsuffix='_3',how= 'outer')
```

[31]:

	state	location	apples	pears	blueberries	strawberries
0	MA	bolton	3.0	0.0	3.0	0.0
1	MA	berlin	2.0	3.0	2.0	3.0
2	VT	boyleston	0.0	7.0	0.0	7.0
3	NH	berlin	1.0	2.0	NaN	NaN
3	ME	berlin	NaN	NaN	1.0	2.0

[32]:
```
# Left join

d7.join(d8.set_index(['state','location']), on=['state','location'],␣
 ↪lsuffix='_3',how= 'left')
```

[32]:

	state	location	apples	pears	blueberries	strawberries
0	MA	bolton	3	0	3.0	0.0
1	MA	berlin	2	3	2.0	3.0
2	VT	boyleston	0	7	0.0	7.0
3	NH	berlin	1	2	NaN	NaN

[33]:
```
# Right join

d7.join(d8.set_index(['state','location']), on=['state','location'],␣
↪lsuffix='_3',how= 'right')
```

[33]:

	state	location	apples	pears	blueberries	strawberries
0	MA	bolton	3.0	0.0	3	0
1	MA	berlin	2.0	3.0	2	3
2	VT	boyleston	0.0	7.0	0	7
3	ME	berlin	NaN	NaN	1	2

2.1 Append (aka union)

Stack datasets on top of one another

[34]:
```
df9 = {
    'month':['Oct','Oct','Oct','Oct'],
    'location':['bolton','berlin','boyleston','charlton'],
    'apples': [3, 2, 0, 1],
    'pears': [0, 3, 7, 2]
}

df10 = {
    'month':['Nov','Nov','Nov','Nov'],
    'location':['bolton','berlin','boyleston','charlton'],
    'apples': [3, 2, 0, 1],
    'pears': [0, 3, 7, 2]
}

d9 = pd.DataFrame(df9)
d10 = pd.DataFrame(df10)
```

[35]:
```
d9.append(d10)
```

[35]:

	month	location	apples	pears
0	Oct	bolton	3	0
1	Oct	berlin	2	3
2	Oct	boyleston	0	7
3	Oct	charlton	1	2
0	Nov	bolton	3	0
1	Nov	berlin	2	3
2	Nov	boyleston	0	7

```
3   Nov   charlton       1      2
```

2.2 Concatenate

With concatenation, your datasets are just stitched together along an axis — either the row axis or column axis

[36]:
```
pd.concat([d9,d10])  # Need to pass in a *list* of dataframes
```

[36]:
```
  month   location  apples  pears
0   Oct     bolton       3      0
1   Oct     berlin       2      3
2   Oct  boyleston       0      7
3   Oct   charlton       1      2
0   Nov     bolton       3      0
1   Nov     berlin       2      3
2   Nov  boyleston       0      7
3   Nov   charlton       1      2
```

Concat can work along either axis

[37]:
```
pd.concat([d9,d10], axis = 1)
```

[37]:
```
  month   location  apples  pears month   location  apples  pears
0   Oct     bolton       3      0   Nov     bolton       3      0
1   Oct     berlin       2      3   Nov     berlin       2      3
2   Oct  boyleston       0      7   Nov  boyleston       0      7
3   Oct   charlton       1      2   Nov   charlton       1      2
```

[38]:
```
pd.concat([d7,d8], axis = 1)  # Be careful! (Look at Berlin output NH & ME)
```

[38]:
```
  state   location  apples  pears state   location  blueberries  strawberries
0    MA     bolton       3      0    MA     bolton            3             0
1    MA     berlin       2      3    MA     berlin            2             3
2    VT  boyleston       0      7    VT  boyleston            0             7
3    NH     berlin       1      2    ME     berlin            1             2
```

[39]:
```
pd.concat([d7,d8], join = 'inner', axis =1)
```

[39]:
```
  state   location  apples  pears state   location  blueberries  strawberries
0    MA     bolton       3      0    MA     bolton            3             0
1    MA     berlin       2      3    MA     berlin            2             3
2    VT  boyleston       0      7    VT  boyleston            0             7
3    NH     berlin       1      2    ME     berlin            1             2
```