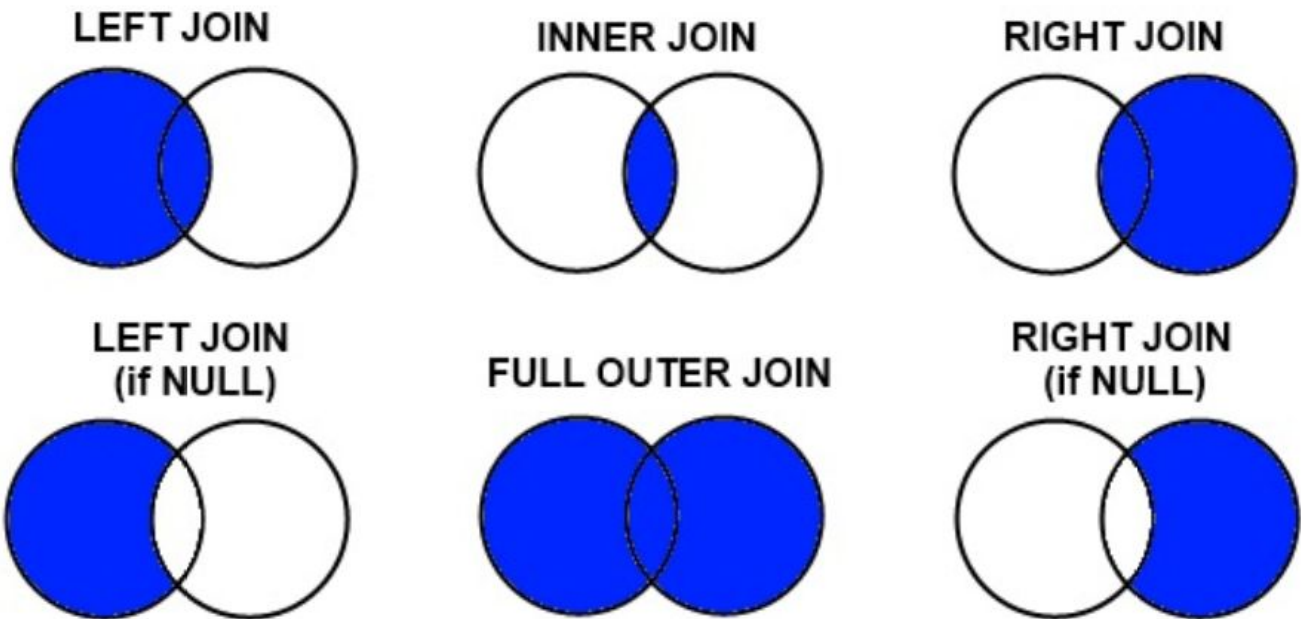


T5 - Data analysis techniques and methodologies



- Merging on Dataframes Columns

We can merge Dataframes N:1 and N:N

pandas.merge(<Dataframe_1>, ..., <Dataframe_n>) -> Looks for strictly coincidences index and labels

pandas.merge(<Dataframes>, on= <ColumnLabels>) -> Label exact coincidence values, not indexes.

pandas.merge(<dfs>, on = <CoLabels>, how= { 'inner', 'right', 'left', 'outer' }) -> Order to merge:

inner: for labels, after indexes df1, ..., after indexes dfn. Default value for merge

right: for Colabels, after labels dfn, ..., after labels df1. NaN not permitted on right dfs labels

left: for CoLabels, after labels df1, ..., after labels dfn. NaN not permitted on left dfs labels

outer: same as inned, but permits NaN for any non combination.

pandas.merge(....., suffixes=<suffix list for labels not in CoLabels>)

- Merging on DataFrames Indexes

Merge index to index -> **left_index = True, right_index = True**

Merge label with index -> left_on = < list of labels>, right_index = True

| -> right_on=< list of labels>, left_index = True

- Joining Dataframes with same indexes

Joining dataframe to other dataframe: adding combinations and columns for items:

Example1: `df1.join(df2)`

df1	data	df2	profit	df1.j(df2)	data	profit
O	0	O	10	L	2	NaN
U	1	O	20	O	0	10.0
L	2	U	20	O	0	20.0
O	3			O	3	10.0
U	4			O	3	20.0
				U	1	20.0
				U	4	20.0

- Concatenation of Series

Concatenate/link them along specific access: **axis = 0** -> rows, **axis = 1** -> columns (generates Dataframe)

For new behaviour (not sort by default) on **axis = 1** -> **sort = False**

Label indexes by **names=**

Set subindexes by **keys=**

```
# Concatenate/link numpy arrays
print '--- Concatenate a1, b1 rows(axis=0)'
print np.concatenate([a1, b1], axis=0)

print '--- Concatenate a1, b1 columns(axis=1)'
print np.concatenate([a1, b1], axis=1)

# Create Series
s1 = Series([100, 200, 300], index=['A', 'B', 'C'])
s2 = Series([400, 500], index=['D', 'E'])
print '--- s1 ---'
print s1
print '--- s2 ---'
print s2

# Concatenate/link Series
print '--- concat(s1,s2) axis = 0---'
print pd.concat([s1, s2])
print '--- concat(s2, s1) axis = 0---'
print pd.concat([s2, s1])
print '--- series concat(s1,s2) axis = 0---'
s = pd.concat([s1, s2], axis=0,
              keys=['s1', 's2'],
              names=['idx_s', 'idx'])
print s

print '--- series concat(s1,s2) axis = 1---'
s = pd.concat([s1, s2], axis=1, sort=False,
              keys=['s1', 's2'],
              names=['idx'])
print s
```

```
--- concat(s1,s2) axis = 0---
A    100
B    200
C    300
D    400
E    500
dtype: int64
--- concat(s2,s1) axis = 0---
D    400
E    500
A    100
B    200
C    300
dtype: int64
--- series concat(s1,s2) axis = 0---
idx_s  idx
s1     A    100
      B    200
      C    300
s2     D    400
      E    500
dtype: int64
--- series concat(s1,s2) axis = 1---
idx    s1    s2
A     100.0  NaN
B     200.0  NaN
C     300.0  NaN
D      NaN  400.0
E      NaN  500.0
```

- Concatenation of dataframes

pandas.concat([df1, df2, axis = 0/1, sort=False, ignore_index=True)

Same as pandas.Series, but for recreate continuous index use **ignore_index=True**.

Cell values not assigned were filled with NaN.

```
print '--- concat(s1,s2) axis = 0---'
print pdconcat([s1, s2])
print '--- concat(s2, s1) axis = 0---'
print pdconcat([s2, s1])
print '--- series concat(s1,s2) axis = 0---'
s = pdconcat([s1, s2], axis=0,
              keys=['s1', 's2'],
              names=['idx_s', 'idx'])
print s

print '--- series concat(s1,s2) axis = 1---'
s = pdconcat([s1, s2], axis=1, sort=False,
              keys=['s1', 's2'],
              names=['idx'])
print s

# Create Dataframes
df1 = DataFrame(random.randn(4, 3),
                 columns=['A', 'B', 'C'])
df2 = DataFrame(random.randn(3, 3),
                 columns=['B', 'D', 'A'])
print '--- df1 ---'
print df1

print '--- df2 ---'
print df2

# Concatenate/link Dataframes
print '--- df concat(df1, df2) axis=0 ---'
print pdconcat([df1, df2], axis=0, sort=False)
print '--- df concat(df1, df2) axis=0 ignore_index ---'
print pdconcat([df1, df2], axis=0, sort=False,
                 ignore_index=True)
print '--- df concat(df1, df2) axis=1 ---'
print pdconcat([df1, df2], axis=1, sort=False)
```

```
--- df1 ---
   A      B      C
0  0.904843  0.013675 -1.758741
1  0.607929  0.753118 -1.182747
2 -0.120983  1.201189  0.981723
3  1.009936  0.680958 -1.351300
--- df2 ---
   B      D      A
0  0.124771 -1.821855 -1.395193
1 -1.247934  2.219757 -1.057643
2 -2.446749  0.557201  0.168668
--- df concat(df1, df2) axis=0 ---
   A      B      C      D
0  0.904843  0.013675 -1.758741  NaN
1  0.607929  0.753118 -1.182747  NaN
2 -0.120983  1.201189  0.981723  NaN
3  1.009936  0.680958 -1.351300  NaN
0 -1.395193  0.124771  NaN -1.821855
1 -1.057643 -1.247934  NaN  2.219757
2  0.168668 -2.446749  NaN  0.557201
--- df concat(df1, df2) axis=0 ignore_index ---
   A      B      C      D
0  0.904843  0.013675 -1.758741  NaN
1  0.607929  0.753118 -1.182747  NaN
2 -0.120983  1.201189  0.981723  NaN
3  1.009936  0.680958 -1.351300  NaN
4 -1.395193  0.124771  NaN -1.821855
5 -1.057643 -1.247934  NaN  2.219757
6  0.168668 -2.446749  NaN  0.557201
--- df concat(df1, df2) axis=1 ---
   A      B      C      B      D      A
0  0.904843  0.013675 -1.758741  0.124771 -1.821855 -1.395193
1  0.607929  0.753118 -1.182747 -1.247934  2.219757 -1.057643
2 -0.120983  1.201189  0.981723 -2.446749  0.557201  0.168668
3  1.009936  0.680958 -1.351300  NaN  NaN  NaN
```

- Combining Series and Dataframes

Example combining series if values are NaN:

```
from numpy import nan, float64, arange, where
from pandas import Series, DataFrame, isnull

s1=Series([5, nan, 6, nan], index=['A', 'B', 'C', 'D'])
print '--- s1 ---'
print s1

s2=Series(arange(4), dtype=float64, index=s1.index)
print '--- s2 ---'
print s2

# Substitution values for NaN with where
# isnull() selects choice s1 or s2 values
s3 = Series(where(isnull(s1), s2, s1), index=s1.index)
print '--- combine s2 values if s1 value is NaN ---'
print s3

# The same with combine_first function
s4 = s1.combine_first(s2)
print '---- Same with combine_first method ----'
print s4
```


```
--- s1 ---
A    5.0
B    NaN
C    6.0
D    NaN
dtype: float64
--- s2 ---
A    0.0
B    1.0
C    2.0
D    3.0
dtype: float64
- combine s2 values if s1 value is NaN -
A    5.0
B    1.0
C    6.0
D    3.0
dtype: float64
---- Same with combine_first method ----
A    5.0
B    1.0
C    6.0
D    3.0
dtype: float64
```

Example combining Dataframes, same method **combine_first**:

```
df1 = DataFrame({'col1': [5, nan, 15],
                  'col2': [20, 25, nan],
                  'col3': [nan, nan, 35]})
print '--- df1 ---'
print df1

df2 = DataFrame({'col1': [0, 10, 15],
                  'col3': [10, 20, 30]})
print '--- df2 ---'
print df2

# Combine Dataframes
print '- df1 combine_first df2'
print df1.combine_first(df2)
```



The image shows a Jupyter Notebook interface with a code editor on the left and a console output on the right. The code defines two DataFrames, df1 and df2, and then uses the combine_first method to combine them. The output shows the individual DataFrames and the result of the combination.

--- df1 ---			
	col1	col2	col3
0	5.0	20.0	NaN
1	NaN	25.0	NaN
2	15.0	NaN	35.0

--- df2 ---		
	col1	col3
0	0	10
1	10	20
2	15	30

- df1 combine_first df2			
	col1	col2	col3
0	5.0	20.0	10.0
1	10.0	25.0	20.0
2	15.0	NaN	35.0