

July-26th

Python-tutorial 14

• filling null values :

- ✓ `df.fillna("MADHU")`
- ✓ `df.fillna(value=df['A'].mean())`
(\therefore mean of 'A' column of 'df')

• grouping :

- ✓ `df.groupby('Company')`
- ✓ `df.groupby('Company').mean()` *any operation*

suppose we have df

	Company	Person	Sales		Sales
0	GOOG	A	200	company	296.5
1	GOOG	B	120	GOOG	160.6
2	MSFT	C	340	MSFT	232.0
3	MSFT	D	124	FB	296.5
4	FB	E	243		
5	FB	F	350		

- ✓ `df.groupby('Company').describe()`

(by default it takes numerical columns)

• transpose dataframe (row \rightarrow column, column \rightarrow row)

`df.transpose()` or `df.T`

- for extracting row inside row (one column inside column):
`df1.loc['Sales'].loc['mean']`

- ^{same} merge / join / concatenation :-

① concatenation along row:

`df = pd.concat([df1, df2, df3], axis=0)`

(∵ df1, df2, df3 are dataframes)

② concatenation along columns:

axis=1

ex: df1:

	A1	B1	C1
0	a	b	c
1	d	e	f
2	g	h	i

df2:

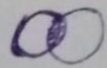
	A2	B1	C1
0	j	k	l
1	m	n	o
2	p	q	r

df3:

	A3	B1	C1
3	s	t	u
4	v	w	x
5	y	z	a

- Concatenate
merge along row:

	A1	B1	C1	A2	B1	C1	A3	B1	C1
0	a	b	c	j	k	l			
1	d	e	f	m	n	o			
2	g	h	i	p	q	r			
3							s	t	u
4							v	w	x
5							y	z	a

merge \rightarrow inner (intersection) 

• concatenate
merge along column:

	A1	A2	A3	B1	B2	B3	C1	C2	C3
0	a			b	c				
1	d	NaN		e	f				
2	g			h	i				
0		j		k	l				
1	NaN	m	NaN	n	o				
2		p		q	r				
3			s	t	u				
4	NaN		v	w	x				
5			y	z	a				

③ merge: (left, right, outer, inner = how) ^{default}

left: \rightarrow

right: \rightarrow

	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3

	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

• merge those rows having same key (index)

pd.merge(left, right, how='inner', on='key')

name can be anything

[right: " right " "

Note:- Merge will try to merge columns.

	Key	A	B	C	D
0	k0	A0	B0	C0	D0
1	k2	A2	B2	C2	D2
2	k3	A3	B3	C3	D3

• When we have different index names:

pd.merge(df1, df2, how='inner',
left_on='key1', right_on='key2')

④ join: (join will pick indexes)

left-join(right)

key	A	B	C	D
k0	A0	B0	C0	D0
k1	A1	B1	NaN	NaN
k2	A2	B2	C2	D2
k3	A3	B3	C3	D3

Q: (1:08) outer

• Use of apply() function:

(manipulation of any column)

df['col10'] = df['col1'] * 10

df

dataframe 'df' with a new column
'col10' where 'col10' = 'col1' * 10

`len()` → finds length.
[`map` : works well w.r.t. a list.
 `apply` : " " " " series/df.

• for typical operations:

```
# define a function:
def test(x):
    if (x > 500):
        return (10 * np.log10(x))
    else:
        return (x/10)
```

```
df['col2'].apply(test)
```

↑
calling test() function

Q. (1:35)