دوره جامع پایتون در یادگیری ماشین

قسمت پنجم، دیتافریم ها

Pandas^2

Datalk.ir
Created by : Ali Arabshahi
Contact us : Linkedin.com/in/mrAliArabshahi

مفهوم دیتافریم ها در پانداس (DataFrames)

دیتافریم ها در پایتون دقیقا مشابه چندین ستون از داده هامون در یک برگه از اکسل هستن. اگه بخوایم یکم دقیق تر بگیم، فرض کنین چندین **سری** که در جلسه قبل باهاشون آشنا شدیم رو در کنار هم بچینیم به این صورت که همه اون سری ها از ایندکس های مشابه و یکسان استفاده کنن. به این می گن **دیتافریم**!

برای شروع، طبق معمول اول پای ثابت کتابخونه ها در برنامه های دیتا ساینتیستا یعنی **پانداس** و **نامپای** رو فراخونی می کنیم. 😎

```
import pandas as pd
import numpy as np
```

حالا یک سری عدد رو توسط توسط نامپای به صورت تصادفی تولید می کنیم.

خیلی نگران کلمه سیید (seed) نباشین، کارش اینه که برای تولید داده های تصادفی در سیستم من و شما از الگوی شماره101 استفاده می کنه برای همین داده هامون یکسان میشه !یه همچین چیزی 🙂


```
from numpy.random import randn
np.random.seed(101)
```

برای ساختن دیتافریم هم مشابه سری ها، اول دیتا، بعد ایندکس ها و در انتها هم نام ستون ها رو وارد می کنیم همون طور که مشاهده می کنین، خروجی یک جدولی مشابه اکسل هست .خیلی پیچیده اش نکنیم !سطر هایی و ستون هایی و دیتا هایی. به همین سادگی !

صفحه بعد 😶

```
df = pd.DataFrame(randn(5,4),index='A B C D E'.split(),columns='W X Y Z'.split())
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

انتخاب ستون دلخواه

فرض کنین میخوایم قسمت مشخصی از دیتافریم رو دریافت کنیم .برای این کار رویکرد های متفاوتی وجود داره که در ادامه با ذکر مثال با همشون آشنا میشیم.

برای انتخاب کردن یک ستون دلخواه می تونیم اسم اون ستون رو داخل براکت صدا بزنیم :

```
df['W']
```

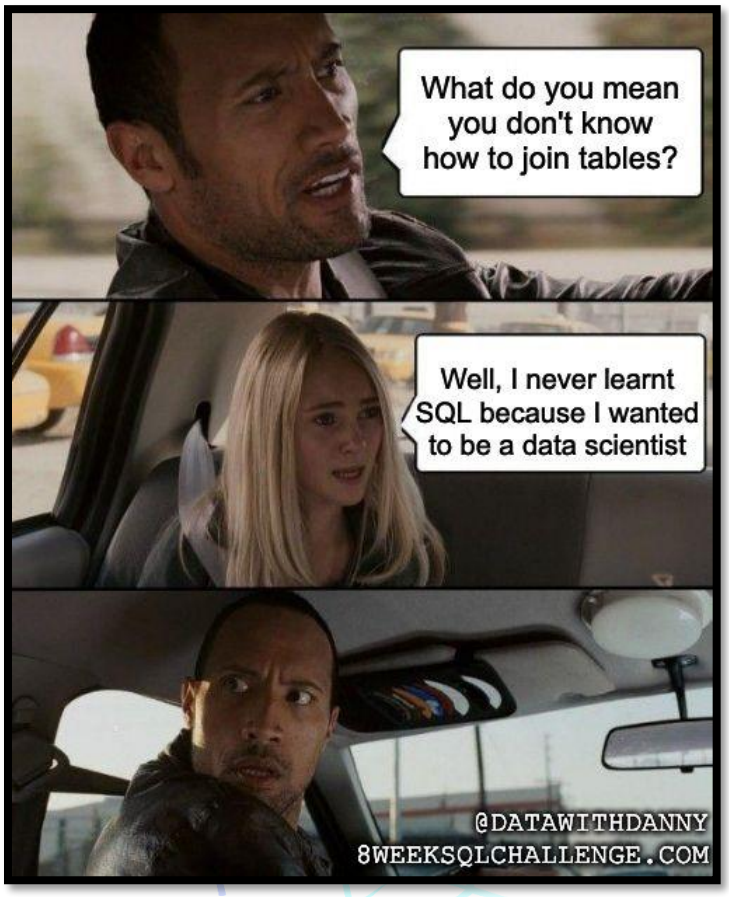
```
A    2.706850
B    0.651118
C   -2.018168
D    0.188695
E    0.190794
Name: W, dtype: float64
```

حالا اگر چند تا ستون رو بخوایم، کافیه فقط اسم ستون های دلخواه رو در قالب یک لیست به پانداس تحویل بدیم

```
df[['W','Z']]
```

	W	Z
A	2.706850	0.503826
B	0.651118	0.605965
C	-2.018168	-0.589001
D	0.188695	0.955057
E	0.190794	0.683509

نمی دونم **با اس کیو ال** کار کردین یا نه !



😀😀😀

مشابه اونجا، در پایتون هم می تونیم یک ستون رو اینطوری صدا بزنیم که البته این روش خیلی پیشنهاد نمیشه :

```
df.W
```

```
A    2.706850
B    0.651118
C   -2.018168
D    0.188695
E    0.190794
Name: W, dtype: float64
```

با دستور زیر می خوایم ببینیم که نوع این ستونی که فراخونی کردیم از چه نوعی هست؟ همون طور که میبینین، یک سری هست !قبلا به این نکته اشاره کرده بودیم که در واقع دیتافریم ها ترکیبی از چند سری هستن که در کنار هم دیگه قرار گرفتن

```
type(df['W'])
```

```
pandas.core.series.Series
```

ساختن یک ستون جدید

برای اضافه کردن یک ستون جدید کافیه ابتدا اسم اون ستون رو بنویسیم و در ادامه مقدار دلخواه رو به ستون جدیدمون اختصاص بدیم.مثلا ستون جدیدی می خوایم که حاصل جمع دو تا ستون زیر باشه:

```
df['new'] = df['W'] + df['Y']
```

	W	X	Y	Z	new
A	2.706850	0.628133	0.907969	0.503826	3.614819
B	0.651118	-0.319318	-0.848077	0.605965	-0.196959
C	-2.018168	0.740122	0.528813	-0.589001	-1.489355
D	0.188695	-0.758872	-0.933237	0.955057	-0.744542
E	0.190794	1.978757	2.605967	0.683509	2.796762

حذف ستون ها

این کار توسط دستور زیر صورت می گیره اما حواسمون به یه نکته باشه .در حالت پیشفرض **اکسیس** ها(axis) برابر صفر هستند .اکسیس !صفر به معنی **سطر** و اکسیس یک به معنی **ستون** هست.در مثال زیر اگر اکسیس برابر صفر در نظر گرفته می شد و یا حتی آورده نمی شد، پانداس اینجور دریافت می کرد که ما یک سطری داریم به اسم نییوو !و می خوایم اون رو حذف کنیم اما چون در دیتافرممون چنین سطری وجود نداره، به ما ارور تحویل می داد !می تونین خودتونم امتحان کنین 😉

```
df.drop('new',axis=1)
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

یه موضوع خیلی مهم !مگه ما در بالا اون ستون رو حذف نکردن؟ پس چرا دوباره ظاهر میشه 🤷

```
df
```

	W	X	Y	Z	new
A	2.706850	0.628133	0.907969	0.503826	3.614819
B	0.651118	-0.319318	-0.848077	0.605965	-0.196959
C	-2.018168	0.740122	0.528813	-0.589001	-1.489355
D	0.188695	-0.758872	-0.933237	0.955057	-0.744542
E	0.190794	1.978757	2.605967	0.683509	2.796762

دلیلش اینه که در اکثر دستور های پانداس برای این که بخوایم تغییرات اعمال شده رو روی دیتافریممون دائمی کنیم باید از دستور اینپلیس !به شرح زیر استفاده کنیم(**inplace**)

```
df.drop('new',axis=1,inplace=True)
```

همون طور که در صفحه بعد می بینین دیگه واقعا از شر اون ستون خلاص شدیم 😊

```
df
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

در پایین هم یک سطر رو حذف کردیم.حتما حالا دیگه می دونین که این تغییر هنوز روی دیتافریم اصلی اعمال نشده.

```
df.drop('E',axis=0)
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057

انتخاب کردن ردیف ها

برای این کار دو رویکرد وجود داره.

رویکرد اول برای زمانی که می خوایم بر اساس نام، ردیفمون رو صدا کنیم .کافیه از دستور زیر استفاده کنیم:

```
df.loc['A']
```

```
W    2.706850
X    0.628133
Y    0.907969
Z    0.503826
Name: A, dtype: float64
```

و رویکرد دوم بر اساس شماره ی اون سطر هست که در مثال زیر سطر شماره دو رو بر اساس این دستور فراخوانی می کنیم. مطمئنم می دونین که بر عکس ما آدم ها! پانداس شمارش سطر ها رو از صفر شروع می کنه پس شماره 2 در اینجا به منزله سطر شماره 3 در واقعیت هست ! 😐

```
df.iloc[2]
```

```
W   -2.018168
X    0.740122
Y    0.528813
Z   -0.589001
Name: C, dtype: float64
```

انتخاب کردن همزمان ردیف ها و ستون ها

برای این کار هم دو رویکرد وجود داره .

معمولا در پایتون زمان هایی که با سطر و ستون سر و کار داشته باشیم اول سطر و سپس ستون آورده می شه. پس در اینجا هم منظور، سطر بی !و ستون وای !هست 🤠


```
df.loc['B','Y']
```

```
-0.8480769834036315
```

حالا اگر چند تا سطر و ستون رو مدنظر قرار داشته باشیم ، ابتدا سطر ها رو در قالب یک لیست و سپس ستون ها رو بر اساس لیست به پانداس تحویل می دیم.

```
df.loc[['A','B'],['W','Y']]
```

	W	Y
A	2.706850	0.907969
B	0.651118	-0.848077

انتخاب شرطی

معمولا خیلی به این برمیخوریم که قسمتی از دیتافرممون رو می خوایم که دارای یک **شرط** یا شروط خاصی هست.

دیتافریمی که تعریف کردیم رو در نظر بگیرین: 🧐

```
df
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

حالا میخوایم مقادیری رو داشته باشیم که از صفر بزرگتر باشن یعنی در شرط زیر صدق کنن:

```
df>0
```

	W	X	Y	Z
A	True	True	True	True
B	True	False	False	True
C	False	True	True	False
D	True	False	False	True
E	True	True	True	True

پس این شرط رو روی دیتافریممون اعمال و سپس فراخوانی می کنیم.

دیتافریم جانه من! از دیتا فریم من!!اونجاهاییش که در این شرط **صدق می کردن** رو به من برگردون !😈

در نتیجه جاهایی که در شرط صدق می کردن، فراخونی می شن و قسمت هایی که در شرط صدق نمی کردن تحت عنوان **نال** یا **بی معنی** نشون داده می شن:

```
df[df>0]
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	NaN	NaN	0.605965
C	NaN	0.740122	0.528813	NaN
D	0.188695	NaN	NaN	0.955057
E	0.190794	1.978757	2.605967	0.683509

حالا فرض‌کنیم مقادیری از دیتا فریممون رو می خوایم که ستون دبلیو !اون، مقادیرش مثبت باشن.یعنی کاری با نال یا بی معنی ها نداریم و فقط مثبت ها رو می خوایم .روش زیر خیلی برای این کار جواب میده 🤠

```
df[df['W']>0]
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

و در نهایت همون چیزی که بالا بهش رسیدیم، ستون وای !اش رو می خوایم .قبول دارم شاید در نگاه اول یکم ترسناک به نظر برسه اما در واقع فقط چند تا کار رو پشت سر هم در یک خط انجام دادیم. 😶

```
df[df['W']>0]['Y']
```

```
A    0.907969
B   -0.848077
D   -0.933237
E    2.605967
Name: Y, dtype: float64
```

برای فراخونی مقادیری که در بیش از یک شرط صدق می کنن، از دستور **اند** (and) استفاده می کنیم.

موضوع مهمی که هست اینه که در موقعیت های اینچینی باید به جای کلمه **and** از حرف **&** استفاده کنیم. دلیل خاص خودش رو داره اما بجاش از علامت زیر استفاده کنین

```
df[(df['W']>0) & (df['Y'] > 1)]
```

	W	X	Y	Z
E	0.190794	1.978757	2.605967	0.683509

و برای **OR**! 🙄 یا **یا** هم از علامت زیر!!!! (|)

```
df[(df['W']>0) | (df['Y'] > 1)]
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

تغییر ایندکس ها

دو حالت وجود داره، یا ایندکسمون رو ریست می کنیم یعنی به حالت پیشفرض که عبارت هست از شماره ی صفر تا ... ریست می کنیم که در این حالت ایندکس های قبلیمون به یک ستون جدید تبدیل می شن

```
df.reset_index()
```

	index	W	X	Y	Z
0	A	2.706850	0.628133	0.907969	0.503826
1	B	0.651118	-0.319318	-0.848077	0.605965
2	C	-2.018168	0.740122	0.528813	-0.589001
3	D	0.188695	-0.758872	-0.933237	0.955057
4	E	0.190794	1.978757	2.605967	0.683509

و در حالت دوم هم یک لیست دلخواه جدید رو می خوایم به عنوان ایندکس های جدیدمون در نظر بگیریم

برای این کار اول لیست رو تعریف می کنیم

```
newind = 'CA NY WY OR CO'.split()
```

و در ادامه اون لیست رو به عنوان یک ستون جدید به دیتافرممون اضافه می کنیم


```
df['States'] = newind
```

	W	X	Y	Z	States
A	2.706850	0.628133	0.907969	0.503826	CA
B	0.651118	-0.319318	-0.848077	0.605965	NY
C	-2.018168	0.740122	0.528813	-0.589001	WY
D	0.188695	-0.758872	-0.933237	0.955057	OR
E	0.190794	1.978757	2.605967	0.683509	CO

و در قدم نهایی اون ستون رو توسط دستور زیر به ایندکس های جدیدمون تبدیل می کنیم.

```
df.set_index('States')
```

States	W	X	Y	Z
CA	2.706850	0.628133	0.907969	0.503826
NY	0.651118	-0.319318	-0.848077	0.605965
WY	-2.018168	0.740122	0.528813	-0.589001
OR	0.188695	-0.758872	-0.933237	0.955057
CO	0.190794	1.978757	2.605967	0.683509

البته در نظر داشته باشین که در حالت پیشفرض این تغییرمون بر روی دیتا فریم اصلی اعمال نمیشه مگر این که از دستور اینپلیس !که قبلا باهاش آشنا شده بودیم، استفاده کنیم (**inplace**)

```
df
```

	W	X	Y	Z	States
A	2.706850	0.628133	0.907969	0.503826	CA
B	0.651118	-0.319318	-0.848077	0.605965	NY
C	-2.018168	0.740122	0.528813	-0.589001	WY
D	0.188695	-0.758872	-0.933237	0.955057	OR
E	0.190794	1.978757	2.605967	0.683509	CO

```
df.set_index('States',inplace=True)
df
```

States	W	X	Y	Z
CA	2.706850	0.628133	0.907969	0.503826
NY	0.651118	-0.319318	-0.848077	0.605965
WY	-2.018168	0.740122	0.528813	-0.589001
OR	0.188695	-0.758872	-0.933237	0.955057
CO	0.190794	1.978757	2.605967	0.683509

ساختار سلسله مراتبی در پانداس

این بخش یه خورده‌بیش از اندازه پیشرفته است و شاید حتی به عنوان دیتا ساینتیست ، خیلی کم باهاش روبه رو بشیم. برای همین **اصلا نیاز نیست** خیلی عمیق بشین .فقط در این حد که بدونین چنین قابلیتی هم وجود داره و هر زمانی که لازم بود بهش رجوع می کنیم 😀😀😀

پس خیلی دقت نکنین !فقط بدونین کد های زیر رو اجرا می کنیم.

```
outside = ['G1','G1','G1','G2','G2','G2']
inside = [1,2,3,1,2,3]
hier_index = list(zip(outside,inside))
hier_index = pd.MultiIndex.from_tuples(hier_index)
```

```
hier_index
```

```
MultiIndex([('G1', 1),
            ('G1', 2),
            ('G1', 3),
            ('G2', 1),
            ('G2', 2),
            ('G2', 3)],
           )
```


تا در نهایت به همچین چیزی برسیم یعنی یک **ساختار سلسله مراتبی** (**Multi Level index**)

```
df = pd.DataFrame(np.random.randn(6,2),index=hier_index,columns=['A','B'])
df
```

		A	B
G1	1	0.302665	1.693723
	2	-1.706086	-1.159119
	3	-0.134841	0.390528
G2	1	0.166905	0.184502
	2	0.807706	0.072960
	3	0.638787	0.329646

در این ساختار در واقع با دو ستون از ایندکس ها مواجه هستیم و یا به عبارت دیگه با تجمیعی از چند دیتافریم که به صورت زیر می تونیم قسمت های خاص از اون رو فراخونی کنیم

```
df.loc['G1']
```

	A	B
1	0.302665	1.693723
2	-1.706086	-1.159119
3	-0.134841	0.390528

```
df.loc['G1'].loc[1]
```

```
A    0.302665
B    1.693723
Name: 1, dtype: float64
```

همون طور که می بینیم، ایندکس های دیتافریمی که ایجاد کردیم ، اسم نداره! 🤷‍♀️ (در ساختار های غیر سلسله مراتبی چون کلا یه دونه ستون ایندکس داشتیم، خیلی هم مهم نبود که اسم داشته باشه ولی اینجا چرا!)


```
df.index.names
```

```
FrozenList([None, None])
```

با دستور زیر می تونیم براشون اسم هم تعریف کنیم

```
df.index.names = ['Group','Num']
```

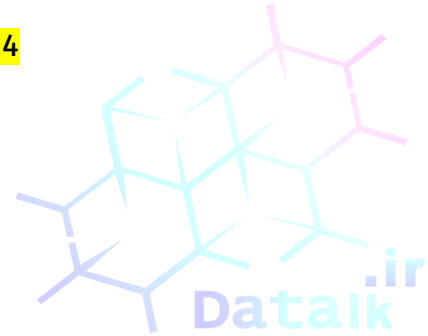
یه متدی هم وجود داره برای یه حالتی خاص؛ فرض کنیم می خوایم از همه ی دیتافریممون با گروه جی وان !ردیف های شماره یک اش رو فراخونی کنیم .متد زیر برای این منظور تعریف شده

```
df
```

		A	B
Group	**Num**		
G1	**1**	0.302665	1.693723
	2	-1.706086	-1.159119
	3	-0.134841	0.390528
G2	**1**	0.166905	0.184502
	2	0.807706	0.072960
	3	0.638787	0.329646

```
df.xs(['G1',1])
```

```
A    0.302665
B    1.693723
Name: (G1, 1), dtype: float64
```




```
df.xs(1,level='Num')
```

	A	B
Group		
G1	0.302665	1.693723
G2	0.166905	0.184502

بازم تاکید می کنم که خیلی نیاز نیست روی این مورد آخری که گفتیم عمیق بشین، فقط در همین حد بدونین که چنین قابلیتی هم وجود داره .خب به پایان این جلسه طولانی اما **بسیار پرکاربرد** رسیدیم، واقعا دمتون گرم

تا جلسه بعدی