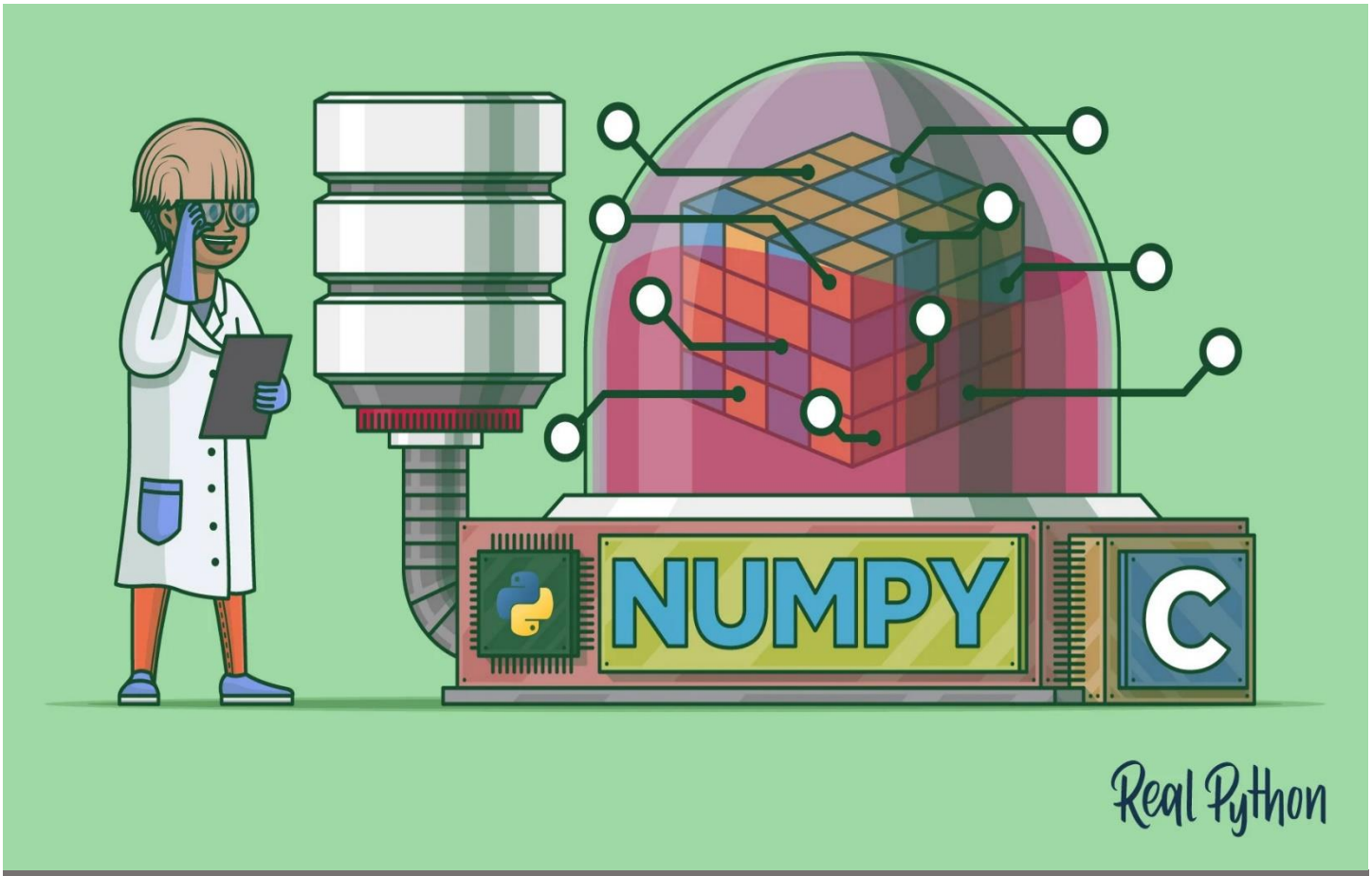

دوره جامع پایتون در یادگیری ماشین

قسمت دوم، انتخاب بخش از آرایه ها

Numpy^2 !

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

انتخاب کردن بخش هایی از آرایه ها در نامپای

در این جلسه به این می پردازیم که چه طوری میشه قسمت های مشخصی از آرایه **یا بردار هامون** رو انتخاب کنیم 🧐

قبل از هر چیز کتابخونه نامپای رو فراخونی می کنیم

```
import numpy as np
```

حالا یه بردار می سازیم

```
arr = np.arange(0,11)
arr
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

انتخاب قسمتی دلخواه از آرایه یک بعدی

ساده ترین راه برای این کار استفاده از براکت هست. توجه کنین که در اینجا با یک آرایه یک بعدی مواجه هستیم. یادتون هست که در لیست های پایتونی چطوری این کار رو انجام می دادیم؟ اینجا هم دقیقا همون طوریه. در پایین هشتمین المان از آرایه مون رو فراخونی می کنیم


```
arr[8]
```

```
8
```

فقط حواستون باشه که دقیقا مشابه لیست ها، اینجا هم شمارش المان ها از صفر انجام میشه. در واقع عدد ۸ ، نهمین المان این آرایه محسوب می شه. اگر باور نمی کنین، بشمارین 😉

در ادامه نیز دوباره شبیه لیست ها می یایم و یک قسمتی از آرایه رو انتخاب می کنیم یعنی از آرایه اول و تا آرایه پنجم. به کلمه **تا** خیلی دقت کنین .تا یعنی نه اون! به این صورت که شامل آرایه اول و غیر شامل عدد دوم! باشه

```
arr[1:5]
```

```
array([2, 3, 4, 5])
```

```
arr[0:5]
```

```
array([1, 2, 3, 4, 5])
```

مفهوم انتشار یا Broadcasting در نامپای

اینجا به یکی از تفاوت های باحال آرایه ها نسبت به لیست ها پی می بریم. در دنیای نامپای ما می تونیم یک تغییر دلخواه رو نسبت که کل آرایه مون بسط بدیم .در مثال زیر اول آرایه های صفرم تا پنجم رو انتخاب می کنیم و بعد همه شون رو تبدیل به ۱۰۰ می کنیم. به همین راحتی!

```
arr[0:5]=100
```

```
arr
```

```
array([100, 100, 100, 100, 100,   6,   7,   8,   9,  10])
```

حالا لیست زیر رو در نظر بگیرین. بیایم و مقادیر صفرم تا دومش رو با هم، دورهمی! تبدیلشون کنیم به ۱۰۰. اما نمیشه! دیدین؟ باز بگین چرا آرایه 😎


```
a=[0,1,2,3,4,5]
```

```
a[0:2]=100
```

😲

```
TypeError                                 Traceback (most recent call last)
<ipython-input-60-1ba9c603c393> in <module>
----> 1 a[0:2]=100
TypeError: can only assign an iterable
```

آرایه مون رو دوباره به شکل اولش در میاریم .

```
arr = np.arange(0,11)
```

```
arr
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

و اما یک **نکته ی خیلی خیلی مهم** در آرایه ها. وقتی ما یک قسمتی از آرایه اصلیمون رو انتخاب می کنیم، مشابه پایین

```
slice_of_arr = arr[0:6]
```

```
slice_of_arr
```

```
array([0, 1, 2, 3, 4, 5])
```

و یک تغییری رو روی اون قسمت خاص اعمال می کنیم، این تغییر نه تنها روی اون قسمت بلکه بر روی آرایه اصلی هم اعمال می شه! این هم از عجایب نامپای هست دیگه، چه می شه کرد. 🤷

```
slice_of_arr[:]=99
```

```
slice_of_arr
```

```
array([99, 99, 99, 99, 99, 99])
```

آرایه اصلی رو مشاهده می کنیم به همراه تغییراتی که روش اعمال شده

```
arr
```

```
array([99, 99, 99, 99, 99, 99,  6,  7,  8,  9, 10])
```

در واقع قسمت هایی از آرایه ها کپی از اون ها نیستن بلکه یه تصویری از آرایه اصلیمون هستن. دلیل این مسئله هم بر می گرده به حافظه و شلوغ نشدن برنامه ها و از این جور مسائل و اما راه حل: توسط دستور زیر اول از آرایه اصلی یه نسخه کپی می گیریم و بعد تغییر رو روی اون نسخه کپی شده اعمال می کنیم تا مطمئن باشیم آرایه اصلیمون بدون تغییر باقی میمونه

```
arr_copy = arr.copy()
```

```
arr_copy
```

```
array([99, 99, 99, 99, 99, 99,  6,  7,  8,  9, 10])
```

انتخاب قسمت دلخواه آرایه دو بعدی

زمانی که با آرایه های دو بعدی یا ماتریس ها سر و کار داریم، در واقع کلید اصلی این دو تا کلمه است. **سطر و ستون**!

هر ماتریسی یک سری سطر یا ردیف داره و یک سری ستون یا فیچر یا ویژگی مثل قد، وزن، سن

آرایه دو بعدی زیر رو در نظر بگیرین

```
arr_2d = np.array(([5,10,15],[20,25,30],[35,40,45]))
```

```
arr_2d
```

```
array([[ 5, 10, 15],
       [20, 25, 30],
       [35, 40, 45]])
```

همیشه داخل براکت عدد اول نشون دهنده اینه که چه سطر هایی رو در نظر داریم و عدد دوم هم بیانگر ستون های دلخواه هست در مثال پایین فقط یک عدد اومده که بیانگر سطر هست و اون هم سطر شماره یک. حتما می دونین که اینجا هم شمارش از صفر شروع میشه و عدد یک بیانگر المان یا سطر شماره دو در واقعیت می باشد

```
arr_2d[1]
```

```
array([20, 25, 30])
```

مثال بعد رو در نظر بگیریم. دو تا عدد تعریف کردیم. گفتیم عدد اول بیانگر سطر و دوم هم بیانگر ستون هست. پس ما المان سطر اول از ستون صفرم رو می خوایم .که در اینجا عدد ۲۰ خواهد بود.

```
arr_2d[1,0]
```

```
20
```

می تونیم برای این کار از دو تا براکت هم استفاده کنیم. هیچ فرقی نمیکنه اما من به شخصه با اولی بیشتر حال می کنم 😋

```
arr_2d[1][0]
```

```
20
```

برای این که بیشتر برامون جا بیفته چند تا مثال دیگه رو هم با هم بررسی می کنیم قبل از ویرگول مربوط به سطر هست که می گه من از سطر شماره صفر تا دو رو می خوام و بعد از ویرگول مربوط به ستون هست که می گه از ستون شماره یک تا ستون آخر که به چنین ماتریسی می رسیم

```
arr_2d[:2,1:]
```

```
array([[10, 15],
       [25, 30]])
```

حالا ردیف دوم

```
arr_2d[2]
```

```
array([35, 40, 45])
```

و حالا ردیف دوم و ستون های اول تا آخر

```
arr_2d[2,:]
```

```
array([35, 40, 45])
```


انتخاب بدون ترتیب

حالا فرض کنیم بدون ترتیب مشخصی قصد انتخاب بعضی از ردیف ها یا ستون ها رو داریم. نامپای برای این کار هم راه حل داره. قبل از هر چیز ماتریس زیر رو تعریف می کنیم.

ببخشید مجبور شدیم بندازیم صفحه بعد 😬

```
arr2d = np.zeros((10,10))
arr_length = arr2d.shape[1]
```

```
for I in range(arr_length):
    arr2d[i] = i
```

```
arr2d
```

```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [2., 2., 2., 2., 2., 2., 2., 2., 2., 2.],
       [3., 3., 3., 3., 3., 3., 3., 3., 3., 3.],
       [4., 4., 4., 4., 4., 4., 4., 4., 4., 4.],
       [5., 5., 5., 5., 5., 5., 5., 5., 5., 5.],
       [6., 6., 6., 6., 6., 6., 6., 6., 6., 6.],
       [7., 7., 7., 7., 7., 7., 7., 7., 7., 7.],
       [8., 8., 8., 8., 8., 8., 8., 8., 8., 8.],
       [9., 9., 9., 9., 9., 9., 9., 9., 9., 9.]])
```

ما ردیف های شماره ۲ و ٤ و ٥ و ٨ رو نیاز داریم. برای این کار قسمت اول که مربوط به سطر ها می شد رو در قالب یک لیست به تابع می دیم و نتیجه می شه همون چیزی که می خواستیم

```
arr2d[[2,4,6,8]]
```

```
array([[2., 2., 2., 2., 2., 2., 2., 2., 2., 2.],
       [4., 4., 4., 4., 4., 4., 4., 4., 4., 4.],
       [6., 6., 6., 6., 6., 6., 6., 6., 6., 6.],
       [8., 8., 8., 8., 8., 8., 8., 8., 8., 8.]])
```

انتخاب شرطی

فرض کنیم می خوایم یه مقادیری که در آرایه دارای یک شرط خاص مثلا بیشتر از ٤ هستند رو فراخونی کنیم. قبل از هر چیز مجددا یه آرایه تعریف میکنیم

```
arr = np.arange(1,11)
arr
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

چیزی که می خوایم دقیقا اینه :آرایه هایی که بزرگتر از ٤ باشن کلا در دنیای پایتون هر شرطی نتیجه اش یا بله هست یا خیر یا به قول پایتون True or False پس در درجه اول پایتون میاد و چک می کنه که برای شرطی که قرار دادین چی صدق می کنه و چی صدق نمیکنه.

```
arr > 4
```

```
array([False, False, False, False,  True,  True,  True,  True,  True,
        True])
```

حالا یک متغیر برای شرطمون تعریف می کنیم یا به قولی شرطمون رو توی یک متغیر می ریزیم! دلیلش رو جلوتر متوجه خواهید شد.

```
bool_arr = arr>4

bool_arr
```

```
array([False, False, False, False,  True,  True,  True,  True,  True,
        True])
```

برای این که فقط مقادیری که در شرط صدق می کنن رو فراخونی کنیم، میایم و می گیم ، آرایه هایی از آرایه اصلی رو به من تحویل بده که آن شرط برایشان تعریف شده بود. شاید در نگاه اول یکم عجیب به نظر برسه اما چند تا مثال که ازش ببینیم خیلی راحت متوجه می شیم:

```
arr[bool_arr]
```

```
array([ 5,  6,  7,  8,  9, 10])
```

```
arr[arr>2]
```

```
array([ 3,  4,  5,  6,  7,  8,  9, 10])
```

```
x = 2
arr[arr>x]
```

```
array([ 3,  4,  5,  6,  7,  8,  9, 10])
```

خدا قوت تا جلسه بعدی