Day_30_251123

January 23, 2024

1 Fitbit data analysis using NumPy 2D arrays

[2]:
```
import numpy as np
```

[1]:
```
!gdown 1vk1Pu0djiYcrdc85yUXZ_Rqq2oZNcohd
```

```
Downloading…
From: https://drive.google.com/uc?id=1vk1Pu0djiYcrdc85yUXZ_Rqq2oZNcohd
To: C:\Data\Data_science\Data Science RIA\3 Python\Codes\fit.txt

  0%|          | 0.00/3.43k [00:00<?, ?B/s]
100%|##########| 3.43k/3.43k [00:00<?, ?B/s]
```

[6]:
```
data = np.loadtxt("fit.txt",dtype='str')
```

[8]:
```
data.ndim
```

[8]:
```
2
```

[9]:
```
data.shape
```

[9]:
```
(96, 6)
```

[11]:
```
date,step_count,mood,calories_burned,hours_of_sleep,activity_status = data.T
```

[13]:
```
step_count = np.array(step_count,dtype='int')
```

[14]:
```
calories_burned = np.array(calories_burned,dtype='int')
```

[15]:
```
hours_of_sleep = np.array(hours_of_sleep,dtype='int')
```

[31]:
```
np.unique(mood,return_counts = 'True')
```

[31]:
```
(array(['Happy', 'Neutral', 'Sad'], dtype='<U10'),
 array([40, 27, 29], dtype=int64))
```

[33]:
```
unique, counts = np.unique(activity_status,return_counts = 'True')
counts
```

[33]:
```
array([42, 54], dtype=int64)
```

Operating with data and getting insights

[18]:
```
step_count.mean()
```

[18]:
```
2935.9375
```

[21]:
```
step_count.max()
```

[21]:
```
7422
```

[23]:
```
step_count.argmax()
```

[23]:
```
69
```

[26]:
```
date[step_count.argmax()]
```

[26]:
```
'14-12-2017'
```

[27]:
```
date[step_count.argmin()]
```

[27]:
```
'08-10-2017'
```

[34]:
```
calories_burned[step_count.argmax()]
```

[34]:
```
243
```

[36]:
```
np.mean(step_count[mood=='Sad'])
```

[36]:
```
2103.0689655172414
```

[38]:
```
np.mean(step_count[mood=='Happy'])
```

[38]:
```
3392.725
```

[39]:
```
np.unique(mood[step_count>4000],return_counts='True')
```

[39]:
```
(array(['Happy', 'Neutral', 'Sad'], dtype='<U10'),
 array([22,  9,  7], dtype=int64))
```

[42]:
```
np.unique(mood[step_count<2000],return_counts='True')
```

[42]:
```
(array(['Happy', 'Neutral', 'Sad'], dtype='<U10'),
 array([13,  8, 18], dtype=int64))
```

[44]:
```
np.mean(hours_of_sleep[activity_status=='Active'])
```

[44]:
```
5.4523809523809526
```

[51]:
```
a = np.arange(9,0,-1).reshape(3,3)
a
```

[51]:
```
array([[9, 8, 7],
       [6, 5, 4],
       [3, 2, 1]])
```

[57]:
```
a.sort(axis = 1)
```

[58]:
```
a
```

[58]:
```
array([[7, 8, 9],
       [4, 5, 6],
       [1, 2, 3]])
```

2 Matrix Multiplications

[12]:
```
a = np.arange(5)
```

To generate the matrix of shape (n*m) with all ones we use np.ones(shape=(n,m))

[70]:
```
b = np.ones(shape=(5)) * 2
a*b
```

[70]:
```
array([0., 2., 4., 6., 8.])
```

[71]:
```
a = np.arange(12).reshape(3,4)
```

[72]:
```
a
```

[72]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

If the matrix shapes are same we do element multiplication a*b or np.dot(a,b) but the shape is Transpose of it we use np.matmul(matrix1,matrix2) or a@b

[85]:
```
a = np.ones(shape=(3,4))
b = np.ones(shape=(4,3))
np.matmul(a,b)
```

[85]:
```
array([[4., 4., 4.],
       [4., 4., 4.],
       [4., 4., 4.]])
```

[86]:
```
c = np.ones(shape=(4,4))
d = np.ones(shape=(4,4))
np.matmul(c,d)
```

[86]:
```
array([[4., 4., 4., 4.],
       [4., 4., 4., 4.],
       [4., 4., 4., 4.],
       [4., 4., 4., 4.]])
```

[87]:
```
c@d
```

[87]:
```
array([[4., 4., 4., 4.],
       [4., 4., 4., 4.],
       [4., 4., 4., 4.],
       [4., 4., 4., 4.]])
```

[101]:
```
f = np.arange(16).reshape(4,4)
g = np.arange(16).reshape(4,4)
np.dot(f,g)
```

[101]:
```
array([[ 56,  62,  68,  74],
       [152, 174, 196, 218],
       [248, 286, 324, 362],
       [344, 398, 452, 506]])
```

[102]:
```
h = np.arange(12).reshape(4,3)
j = np.arange(12).reshape(3,4)
np.matmul(h,j)
```

[102]:
```
array([[ 20,  23,  26,  29],
       [ 56,  68,  80,  92],
       [ 92, 113, 134, 155],
       [128, 158, 188, 218]])
```

3 Vectorization

[3]:
```
z = np.arange(12)
```

[4]:
```
import math
```

[5]:
```
z
```

[5]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

[ ]:
```
x = np.vectorize(math.log)(a)
```

[ ]:
```
x
```

4 3D

[117]:
```
a = np.arange(24).reshape(2,3,4)
```

[118]:
```
a
```

[118]:
```
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]],

       [[12, 13, 14, 15],
        [16, 17, 18, 19],
        [20, 21, 22, 23]]])
```

[114]:
```
a.size
```

[114]:
```
24
```

[120]:
```
a.ndim
```

[120]:
```
3
```

[133]:
```
a = np.arange(12).reshape(3,4)
```

[134]:
```
a
```

[134]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

[135]:
```
b = np.arange(16).reshape(4,4)
```

[136]:
```
b
```

[136]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

[137]:
```
np.matmul(a,b)
```

[137]:
```
array([[ 56,  62,  68,  74],
       [152, 174, 196, 218],
       [248, 286, 324, 362]])
```