Pandas

March 2, 2023

1 Pandas

Pandas adalah library python yang digunakan untuk mengerjakan datasets. Biasanya fungsinya untuk menganalisis, membersihkan, mengeksplore dan mamanipulasi data.

Di bab sebelumnya, kita membahas detail tentang NumPy dan objek ndarray-nya, yang menyediakan penyimpanan efisien dan manipulasi array bertipe padat dengan Python. Di sini kita akan mengembangkan pengetahuan ini dengan melihat secara mendetail pada struktur data yang disediakan oleh library Pandas. Pandas adalah paket baru yang dibangun di atas NumPy, dan menyediakan implementasi DataFrame yang efisien. DataFrames pada dasarnya adalah array multidimensi dengan label baris dan kolom terlampir, dan seringkali dengan tipe heterogen dan/atau data yang hilang.

1.1 Installing and Using Pandas

Menginstal Pandas di sistem Anda memerlukan NumPy untuk diinstal, dan jika Anda sedang membangun library dari sumber, memerlukan alat yang sesuai untuk mengkompilasi sumber C dan Cython tempat Panda dibangun. Detail tentang instalasi ini dapat ditemukan di dokumentasi Panda. Jika Anda mengikuti saran yang diuraikan di kata pengantar dan menggunakan tumpukan Anaconda, Anda sudah menginstal Pandas. Setelah Panda diinstal, Anda dapat mengimpornya dan memeriksa versinya:

[2]:
```
import pandas
pandas.__version__
```

[2]:
```
'1.2.4'
```

Anda bisa mengimport pandas sebagai pd

[3]:
```
 import pandas as pd
```

Ada tiga dasar Pandas data structure: 1. Series 2. DataFrame 3. index

1.2 Pandas Series Object

Pandas series adalah sebuah array satu dimensi yang terindeks. Pandas series terbentuk dari list atau array sebagai berikut:

[4]:
```
import pandas as pd
import numpy as np
```

[5]:
```
data = pd.Series([0.25,0.5,0.75,1.0])
data
```

[5]:
```
0    0.25
1    0.50
2    0.75
3    1.00
dtype: float64
```

Seperti yang kita lihat di keluaran sebelumnya, series membungkus urutan nilai dan urutan indeks, yang dapat kita akses dengan nilai dan atribut indeks. Nilainya hanyalah array NumPy yang sudah dikenal:

[6]:
```
data.values
```

[6]:
```
array([0.25, 0.5 , 0.75, 1.  ])
```

Indeks adalah objek mirip-array bertipe pd.Index, yang akan kita bahas lebih detail:

[7]:
```
data.index
```

[7]:
```
RangeIndex(start=0, stop=4, step=1)
```

Seperti dengan array NumPy, data dapat diakses oleh indeks terkait melalui notasi kurung persegi Python yang sudah dikenal:

[8]:
```
data[1]
```

[8]:
```
0.5
```

[9]:
```
data[1:3]
```

[9]:
```
1    0.50
2    0.75
dtype: float64
```

Namun, seperti yang akan kita lihat, Seri Panda jauh lebih umum dan fleksibel daripada larik NumPy satu dimensi yang ditirunya.

1.3 Pandas DataFrame Object

Struktur fundamental berikutnya di Pandas adalah DataFrame. Seperti objek Seri yang dibahas di bagian sebelumnya, DataFrame dapat dianggap sebagai generalisasi dari array NumPy, atau sebagai spesialisasi dictionary Python. Sekarang kita akan melihat masing-masing perspektif ini.

Jika Series adalah analog dari array satu dimensi dengan indeks fleksibel, DataFrame adalah analog array dua dimensi dengan indeks baris fleksibel dan nama kolom fleksibel. Sama seperti Anda mungkin menganggap array dua dimensi sebagai urutan berurutan dari kolom satu dimensi yang selaras, Anda dapat menganggap DataFrame sebagai urutan objek series yang terurut. Di sini, dengan "disejajarkan" kami maksudkan bahwa mereka berbagi indeks yang sama.

Untuk mendemonstrasikan ini, pertama-tama mari kita buat Series baru yang mencantumkan luas masing-masing dari lima negara bagian yang dibahas di bagian sebelumnya:

[10]:
```
area_dict = {'California': 423967, 'Texas': 695662, 'New York': 141297,
                 'Florida': 170312, 'Illinois': 149995}
area = pd.Series(area_dict)
area
```

[10]:
```
California    423967
Texas         695662
New York      141297
Florida       170312
Illinois      149995
dtype: int64
```

Sekarang kita memiliki ini bersama dengan series populasi, kita dapat menggunakan dictionary untuk membuat objek dua dimensi tunggal yang berisi informasi ini:

[11]:
```
population_dict = {'California': 38332521,
                                'Texas': 26448193,
                                'New York': 19651127,
                                'Florida': 19552860,
                                'Illinois': 12882135}
population = pd.Series(population_dict)
population
```

[11]:
```
California    38332521
Texas         26448193
New York      19651127
Florida       19552860
Illinois      12882135
dtype: int64
```

[12]:
```
states = pd.DataFrame({'population': population,
                                    'area': area})
states
```

[12]:
```
            population    area
California    38332521  423967
Texas         26448193  695662
New York      19651127  141297
Florida       19552860  170312
Illinois      12882135  149995
```

[13]:
```
states.index
```

[13]:
```
Index(['California', 'Texas', 'New York', 'Florida', 'Illinois'],
dtype='object')
```

Selain itu, DataFrame memiliki atribut kolom, yang merupakan penampung objek Indeks label kolom:

[14]:
```
states.columns
```

[14]:
```
Index(['population', 'area'], dtype='object')
```

Jadi DataFrame dapat dianggap sebagai generalisasi dari array NumPy dua dimensi, di mana baris dan kolom memiliki indeks umum untuk mengakses data.

Menyusun DataFrame objects Pandas DataFrame dapat dibangun dengan berbagai cara. Di sini kami akan memberikan beberapa contoh. Hal ini perlu diperhatikan ketika kita membuat mengumpulkan data dan mengolahnya untuk dataset

Dari sebuah single Series object. DataFrame adalah kumpulan objek Series, dan DataFrame satu kolom dapat dibangun dari satu Series:

[15]:
```
pd.DataFrame(population, columns=['population'])
```

[15]:
```
            population
California    38332521
Texas         26448193
New York      19651127
Florida       19552860
Illinois      12882135
```

Dari beberapa single Series object. DataFrame adalah kumpulan objek Series, dan DataFrame satu kolom dapat dibangun dari satu Series:

[27]:
```
# Create Series by assigning names
courses = pd.Series(["Spark","PySpark","Hadoop"], name='courses')
fees = pd.Series([22000,25000,23000], name='fees')
discount  = pd.Series([1000,2300,1000],name='discount')

df=pd.concat([courses,fees,discount],axis=1)
print(df)
```

```
   courses   fees  discount
0    Spark  22000      1000
1  PySpark  25000      2300
2   Hadoop  23000      1000
```

[28]:
```
df.columns
```

[28]:
```
Index(['courses', 'fees', 'discount'], dtype='object')
```

Dari sebuh dua dimensi NumPy array. Diberikan array data dua dimensi, kita dapat membuat DataFrame dengan nama kolom dan indeks yang ditentukan. Jika dihilangkan, indeks bilangan bulat akan digunakan untuk masing-masing:

[29]:
```
pd.DataFrame(np.random.rand(3, 2),
                          columns=['foo', 'bar'],
                          index=['a', 'b', 'c'])
```

[29]:

	foo	bar
a	0.887890	0.323803
b	0.903336	0.311701
c	0.148782	0.174436

[31]:
```
#Tanpa kolom
pd.DataFrame(np.random.rand(3, 2))
```

[31]:

	0	1
0	0.728830	0.322371
1	0.212718	0.913191
2	0.873863	0.258020

Dari NumPy array terstrukture .Pandas DataFrame beroperasi seperti array terstruktur, dan dapat dibuat langsung dari satu array:

[34]:
```
A = np.zeros(3, dtype=[('A', 'i8'), ('B', 'f8')])
A
```

[34]:
```
array([(0, 0.), (0, 0.), (0, 0.)], dtype=[('A', '<i8'), ('B', '<f8')])
```

[35]:
```
pd.DataFrame(A)
```

[35]:

	A	B
0	0	0.0
1	0	0.0
2	0	0.0

Contoh Menyusun DataFrame objects Sensor inersia. Berikut akuisisi data dari sensor akselerometer. Ada 3 array dari 3 akuisisi data dari sensor imu: AX (acceleration X),AY (acceleration Y), AZ (acceleration Z). Data tersebut digabungkan menjadi dataframe. Berikut contohnya

[59]:
```
AX = pd.Series([-9.533711010742184,-0.6440402465820312,0.1867477294921875,7.
↪183206106870228,1.1908396946564883,0.4122137404580153,9], name='AX')
AY = pd.Series([-9.593566052246091,-0.5291185668945312,-0.03591302490234375,1.
↪1450381679389312,-4.8473282442748085,0.33587786259541985,9], name='AY')
AZ = pd.Series([-9.447519750976562,-0.5506663818359374,0.033518823242187495,5.
↪923664122137405,5.0458015267175576,1.1145038167938932,9], name='AZ')

df_Acc=pd.concat([AX,AY,AZ],axis=1)
print(df_Acc)
```

```
         AX        AY        AZ
0 -9.533711 -9.593566 -9.447520
```

```
1 -0.644040 -0.529119 -0.550666
2  0.186748 -0.035913  0.033519
3  7.183206  1.145038  5.923664
4  1.190840 -4.847328  5.045802
5  0.412214  0.335878  1.114504
6  9.000000  9.000000  9.000000
```

[56]:
```
df_Acc['label'] = [1,1,2,2,0,0,0]
df_Acc
```

[56]:
```
          AX        AY        AZ  label
0 -9.533711 -9.593566 -9.447520      1
1 -0.644040 -0.529119 -0.550666      1
2  0.186748 -0.035913  0.033519      2
3  7.183206  1.145038  5.923664      2
4  1.190840 -4.847328  5.045802      0
5  0.412214  0.335878  1.114504      0
6  9.000000  9.000000  9.000000      0
```

[57]:
```
AX = pd.Series([-9.533711010742184,-0.6440402465820312,0.1867477294921875,7.
 ↪183206106870228,1.1908396946564883,0.4122137404580153,9], name='AX')
AY = pd.Series([-9.593566052246091,-0.5291185668945312,-0.03591302490234375,1.
 ↪1450381679389312,-4.8473282442748085,0.33587786259541985,9], name='AY')
AZ = pd.Series([-9.447519750976562,-0.5506663818359374,0.033518823242187495,5.
 ↪923664122137405,5.0458015267175576,1.1145038167938932,9], name='AZ')
label =  pd.Series( [1,1,2,2,0,0,0], name='label')
df_Acc_additional=pd.concat([AX,AY,AZ,label],axis=1)
print(df_Acc_additional)
```

```
         AX        AY        AZ  label
0 -9.533711 -9.593566 -9.447520      1
1 -0.644040 -0.529119 -0.550666      1
2  0.186748 -0.035913  0.033519      2
3  7.183206  1.145038  5.923664      2
4  1.190840 -4.847328  5.045802      0
5  0.412214  0.335878  1.114504      0
6  9.000000  9.000000  9.000000      0
```

[58]:
```
df_Acc = df_Acc.append(df_Acc_additional, ignore_index=True)
df_Acc
```

[58]:
```
           AX        AY        AZ  label
0  -9.533711 -9.593566 -9.447520      1
1  -0.644040 -0.529119 -0.550666      1
2   0.186748 -0.035913  0.033519      2
3   7.183206  1.145038  5.923664      2
4   1.190840 -4.847328  5.045802      0
5   0.412214  0.335878  1.114504      0
```

```
6   9.000000  9.000000  9.000000      0
7  -9.533711 -9.593566 -9.447520      1
8  -0.644040 -0.529119 -0.550666      1
9   0.186748 -0.035913  0.033519      2
10  7.183206  1.145038  5.923664      2
11  1.190840 -4.847328  5.045802      0
12  0.412214  0.335878  1.114504      0
13  9.000000  9.000000  9.000000      0
```

[79]:
```
# List1
lst = [[-9.533711010742184,-0.6440402465820312,0.1867477294921875], [-9.
 ↪593566052246091,-0.5291185668945312,-0.03591302490234375],
       [-9.447519750976562,-0.5506663818359374,0.03351882324218749]]

# creating df object with columns specified
df_Acc = pd.DataFrame(lst, columns =['AX', 'AY','AZ'])

print(df_acc )
```

```
         AX        AY        AZ
0 -9.533711 -0.644040  0.186748
1 -9.593566 -0.529119 -0.035913
2 -9.447520 -0.550666  0.033519
```

[80]:
```
df_Acc['label'] = [1,1,2]
df_Acc
```

[80]:
```
          AX        AY        AZ  label
0 -9.533711 -0.644040  0.186748      1
1 -9.593566 -0.529119 -0.035913      1
2 -9.447520 -0.550666  0.033519      2
```

[81]:
```
df_Acc = df_Acc.append(df_Acc, ignore_index=True)
df_Acc
```

[81]:
```
          AX        AY        AZ  label
0 -9.533711 -0.644040  0.186748      1
1 -9.593566 -0.529119 -0.035913      1
2 -9.447520 -0.550666  0.033519      2
3 -9.533711 -0.644040  0.186748      1
4 -9.593566 -0.529119 -0.035913      1
5 -9.447520 -0.550666  0.033519      2
```

1.4 Data Selection

Data Selection untuk data frame. Pada pengolahan data pada mechine learning, biasanya data akan diolah dalam bentuk dataframe. Oleh karena itu pada hands on ini hanya dibahas mengenai data selection DataFrame. Dengan DataFrame contoh sebelumnya kita akan mencoba untuk

memilih spesifik kolom:

[82]:
```
df_Acc
```

[82]:
```
          AX        AY        AZ  label
0 -9.533711 -0.644040  0.186748      1
1 -9.593566 -0.529119 -0.035913      1
2 -9.447520 -0.550666  0.033519      2
3 -9.533711 -0.644040  0.186748      1
4 -9.593566 -0.529119 -0.035913      1
5 -9.447520 -0.550666  0.033519      2
```

Memilih spesifik kolom

[83]:
```
df_Acc['AX']
```

[83]:
```
0   -9.533711
1   -9.593566
2   -9.447520
3   -9.533711
4   -9.593566
5   -9.447520
Name: AX, dtype: float64
```

[84]:
```
df_Acc.AX
```

[84]:
```
0   -9.533711
1   -9.593566
2   -9.447520
3   -9.533711
4   -9.593566
5   -9.447520
Name: AX, dtype: float64
```

Seperti objek Series yang dibahas sebelumnya, sintaks sintak gaya dictionary juga dapat digunakan untuk memodifikasi objek, dalam hal ini untuk menambahkan kolom baru:

Pada contoh adalah formula dari magnitude percepatan formula di 3 dimensi:

[85]:
```
df_Acc.insert(3, "Magnitute", np.square(np.power(df_Acc.AX,2)+np.power(df_Acc.
 ↪AY,2)+np.power(df_Acc.AZ,2)), True)
df_Acc
```

[85]:
```
          AX        AY        AZ    Magnitute  label
0 -9.533711 -0.644040  0.186748  8343.234582      1
1 -9.593566 -0.529119 -0.035913  8522.569883      1
2 -9.447520 -0.550666  0.033519  8020.991168      2
3 -9.533711 -0.644040  0.186748  8343.234582      1
4 -9.593566 -0.529119 -0.035913  8522.569883      1
```

```
5 -9.447520 -0.550666  0.033519  8020.991168      2
```

Seperti disebutkan sebelumnya, kita juga dapat melihat DataFrame sebagai array dua dimensi yang disempurnakan. Kita dapat memeriksa larik data mentah yang mendasarinya menggunakan atribut nilai:

[88]:
```
df_Acc.values
```

[88]:
```
array([[-9.53371101e+00, -6.44040247e-01,  1.86747729e-01,
         8.34323458e+03,  1.00000000e+00],
       [-9.59356605e+00, -5.29118567e-01, -3.59130249e-02,
         8.52256988e+03,  1.00000000e+00],
       [-9.44751975e+00, -5.50666382e-01,  3.35188232e-02,
         8.02099117e+03,  2.00000000e+00],
       [-9.53371101e+00, -6.44040247e-01,  1.86747729e-01,
         8.34323458e+03,  1.00000000e+00],
       [-9.59356605e+00, -5.29118567e-01, -3.59130249e-02,
         8.52256988e+03,  1.00000000e+00],
       [-9.44751975e+00, -5.50666382e-01,  3.35188232e-02,
         8.02099117e+03,  2.00000000e+00]])
```

1.5 Pemilihan Spesifik kolom dan baris untuk data pada machine learnig

Jadi untuk pengindeksan gaya array, kita membutuhkan konvensi lain. Di sini Panda sekali lagi menggunakan pengindeks loc dan iloc. Menggunakan pengindeks iloc, kita dapat mengindeks array yang mendasari seolah-olah itu adalah array NumPy sederhana (menggunakan indeks gaya Python implisit), tetapi indeks DataFrame dan label kolom dipertahankan dalam hasil dibawah ini. Disini akan dicontohkan pengambilan spesifik kolom untuk fitur dan label

[98]:
```
df_Acc
```

[98]:
```
          AX        AY        AZ    Magnitute  label
0 -9.533711 -0.644040  0.186748  8343.234582      1
1 -9.593566 -0.529119 -0.035913  8522.569883      1
2 -9.447520 -0.550666  0.033519  8020.991168      2
3 -9.533711 -0.644040  0.186748  8343.234582      1
4 -9.593566 -0.529119 -0.035913  8522.569883      1
5 -9.447520 -0.550666  0.033519  8020.991168      2
```

iloc

[99]:
```
Feature = df_Acc.iloc[:,:4]
Feature
```

[99]:
```
          AX        AY        AZ    Magnitute
0 -9.533711 -0.644040  0.186748  8343.234582
1 -9.593566 -0.529119 -0.035913  8522.569883
2 -9.447520 -0.550666  0.033519  8020.991168
3 -9.533711 -0.644040  0.186748  8343.234582
```

```
4 -9.593566 -0.529119 -0.035913  8522.569883
5 -9.447520 -0.550666  0.033519  8020.991168
```

[100]:
```
Feature = df_Acc.iloc[:,:-1]
Feature
```

[100]:
```
          AX        AY        AZ    Magnitute
0 -9.533711 -0.644040  0.186748  8343.234582
1 -9.593566 -0.529119 -0.035913  8522.569883
2 -9.447520 -0.550666  0.033519  8020.991168
3 -9.533711 -0.644040  0.186748  8343.234582
4 -9.593566 -0.529119 -0.035913  8522.569883
5 -9.447520 -0.550666  0.033519  8020.991168
```

[101]:
```
Y = df_Acc.iloc[:,-1:]
Y
```

[101]:
```
   label
0      1
1      1
2      2
3      1
4      1
5      2
```

[102]:
```
Y = df_Acc.iloc[:,[4]]
Y
```

[102]:
```
   label
0      1
1      1
2      2
3      1
4      1
5      2
```

loc

[104]:
```
Feature = df_Acc.loc[:,'AX':'Magnitute']
Feature
```

[104]:
```
          AX        AY        AZ    Magnitute
0 -9.533711 -0.644040  0.186748  8343.234582
1 -9.593566 -0.529119 -0.035913  8522.569883
2 -9.447520 -0.550666  0.033519  8020.991168
3 -9.533711 -0.644040  0.186748  8343.234582
4 -9.593566 -0.529119 -0.035913  8522.569883
5 -9.447520 -0.550666  0.033519  8020.991168
```

[109]:
```
Y = df_Acc.loc[:,'label']
Y
```

[109]:
```
0    1
1    1
2    2
3    1
4    1
5    2
Name: label, dtype: int64
```

Gabungan index dan header

[111]:
```
Feature = df_Acc.loc[:3,:'Magnitute']
Feature
```

[111]:

	AX	AY	AZ	Magnitute
0	-9.533711	-0.644040	0.186748	8343.234582
1	-9.593566	-0.529119	-0.035913	8522.569883
2	-9.447520	-0.550666	0.033519	8020.991168
3	-9.533711	-0.644040	0.186748	8343.234582