THỰC HÀNH 2 : NUMPY & PANDAS

Sau bài thực hành này, sinh viên có khả năng:

- Sử dụng thư viện Numpy để xử lý array và ma trận lớn
- Sử dụng thư viện Pandas để xử lý, tính toán dữ liệu

1. GIỚI THIỆU NUMPY

- Numpy là thư viện dành cho lập trình toán học, khoa học, kỹ thuật và khoa học dữ liệu.
- Numpy xử lý rất tốt các đối tượng mảng đa chiều và ma trận.
- Numpy xử lý khối lượng lớn dữ liệu rất nhanh
- Tensorflow và Scikit sử dụng Numpy để tính toán ma trận ở backend.

2. CÀI ĐẶT NUMPY

```
pip install numpy
```

3. TẠO NUMPY ARRAY

3.1. Tạo array 1-chiều

```
#cách 1
# tạo list
my_list = [1, 9, 15, 39]
print('Python list:', my_list)
#chuyển list --> Numpy array
my_numpy_array = np.array(my_list)
print('Numpy array:', my_numpy_array)

#cách 2
a = np.array([1, 9, 15, 39])
print(a)
```

```
Python list: [1, 9, 15, 39]
Numpy array: [ 1  9 15 39]
[ 1  9 15 39]
```

Ta có thể dùng thuộc tính shape, dtype để xem kích thước mảng, kiểu dữ liệu.

3.2. Array 2-chiều

```
c = np.array([(1, 2, 3),
              (4, 5, 6)
              ])
print(c)
print(c.shape)
print(c.dtype)
```

```
[[1 2 3]
 [4 5 6]]
(2, 3)
int64
```

3.3. Array 3-chiều

```
d = np.array([
            [[1., 2., 3.], [4., 5., 6.]],
            [[7., 8., 9.], [10, 11, 12]]
          ])
print(d)
print(d.shape)
```

```
[[[ 1.  2.  3.]
  [ 4.  5.  6.]]

 [[ 7.  8.  9.]
  [10. 11. 12.]]]
(2, 2, 3)
```

3.4. Tạo Array với giá trị ban đầu là 0 hoặc 1

```
e = np.zeros((2, 2))
f = np.ones((3, 3))
print('Zero matrix:\n', e)
print('One matrix\n', f)
```

```
Zero matrix:
 [[0. 0.]
 [0. 0.]]
One matrix
 [[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

3.5. Reshape và Flatten ma trận

3.5.1. Reshape thay đổi kích thước ma trận

```
e = np.array([[1., 2., 3.], [4., 5., 6.]])
print(e)
print(e.shape)
f = e.reshape(3,2)
print(f)
g = np.array([[[1., 2., 3.], [4., 5., 6.]],
              [[7., 8., 9.], [10., 11., 12.]]
              ])
print(g)
print(g.shape)
h = g.reshape(3,4)
print('reshape (3, 4)\n',h)
j = g.reshape(3, 2, 2)
print('reshape (3, 2, 2)\n', j)
k = g.reshape(2, 3, 2)
print('reshape (2, 3, 2)\n', k)
```

```
[[1. 2. 3.]
 [4. 5. 6.]]
(2, 3)
[[1. 2.]
 [3. 4.]
 [5. 6.]]
[[[ 1.  2.  3.]
  [ 4.  5.  6.]]

 [[ 7.  8.  9.]
  [10. 11. 12.]]]
(2, 2, 3)
reshape (3, 4)
 [[ 1.  2.  3.  4.]
 [ 5.  6.  7.  8.]
 [ 9. 10. 11. 12.]]
reshape (3, 2, 2)
 [[[ 1.  2.]
  [ 3.  4.]]

 [[ 5.  6.]
  [ 7.  8.]]

 [[ 9. 10.]
  [11. 12.]]]
```

3.5.2. Platten là chuyển array nhiều chiều về 1 chiều

```
e = np.array([[1., 2., 3.], [4., 5., 6.]])
f = e.flatten()
print(f)
g = np.array([[[1., 2., 3.], [4., 5., 6.]],
              [[7., 8., 9.], [10., 11., 12.]]
              ])
j = g.flatten()
print(j)
```

```
[1. 2. 3. 4. 5. 6.]
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12.]
```

3.6. Thêm dòng dữ liệu với hstack và vstack

Hàm **hstack((x, y))** cho phép thêm nối thêm dữ liệu ma trận y vào sau ma trận x (chiều ngang).

Hàm **vstack((x, y))** cho phép thêm nối thêm dữ liệu ma trận y vào dưới ma trận x (chiều dọc).

```
f = np.array([1., 2., 3.])
g = np.array([4., 5., 6.])
h_stack = np.hstack((f, g))
print('Horizontal Append\n', h_stack)
v_stack = np.vstack((f, g))
print('Vertical Append\n', v_stack)
```

```
Horizontal Append
 [1. 2. 3. 4. 5. 6.]
Vertical Append
 [[1. 2. 3.]
 [4. 5. 6.]]
```

3.7. Sinh mảng số ngẫu nhiên

3.7.1. Sinh mảng số nguyên ngẫu nhiên

```
a = np.random.randint(100, size=(5,))
print('Random 1-D array\n',a)
b = np.random.randint(100, size=(3, 4))
print('Random 2-D array\n',b)
c = np.random.randint(100, size=(3, 4, 3))
print('Random 3-D array\n',c)
```

```
Random 1-D array
 [ 2 75 62 94 66]
Random 2-D array
 [[98 92 94 97]
 [98 69 51 51]
 [50  9 48 84]]
Random 3-D array
 [[[58 40 14]
  [57 42 26]
  [16  9 41]
  [59 84 69]]

 [[62 23  7]
  [97  7 49]
  [36 80 51]
  [83 37 70]]

 [[ 5 86 12]
  [42  3 98]
  [79 23 12]
  [ 1 34 44]]]
```

3.7.2. Sinh mảng số thực trong khoảng [0, 1]

```
#sinh ngẫu nhiên mảng số thực [0, 1]
a = np.random.rand(5)
print('Random 1-D array\n',a)
b = np.random.rand(3, 4)
print('Random 2-D array\n',b)
c = np.random.rand(3, 4, 3)
print('Random 3-D array\n',c)
```

```
Random 1-D array
 [0.97694795 0.58135609 0.37310049 0.04317114 0.95870681]
Random 2-D array
 [[0.20065188 0.77415168 0.01271068 0.54287947]
 [0.08131005 0.23991586 0.76875529 0.33362021]
 [0.47339329 0.53452209 0.41521402 0.81569097]]
Random 3-D array
 [[[0.45141181 0.80935643 0.9196085 ]
  [0.17274109 0.05948666 0.01011627]
  [0.08212584 0.88883691 0.30653759]
  [0.46800835 0.24948332 0.36816523]]

 [[0.31344616 0.52413104 0.87969277]
  [0.80528164 0.06993265 0.67356058]
  [0.82108591 0.58362639 0.01413421]
  [0.69432265 0.67462178 0.89112465]]

 [[0.96110129 0.2429993  0.2232738 ]
  [0.69637372 0.72374122 0.63525499]
  [0.06367548 0.87673229 0.18913645]
  [0.2968745  0.50298948 0.15396021]]]
```

3.7.3. Sinh mảng số ngẫu nhiên theo phân phối chuẩn Gaussian

Ta dùng hàm

Numpy.random.normal(loc, scale, size)

Trong đó

Loc: giá trị trung tâm

Scale: độ lệch chuẩn

Size: số phần tử trả về

```
a = np.random.normal(5, 0.5, 10)
print(a)
```

```
[4.94284102 5.0238746  4.69654707 4.59024079 5.34002763 5.35437726
 4.87984648 5.38341356 4.92519622 4.95704346]
```

3.7.4. Sinh ngẫu nhiên một giá trị từ mảng giá trị cho trước

```
#sinh 1 giá trị ngẫu nhiên từ mảng giá trị cho trước
a = np.random.choice([2, 4, 6, 8, 10, 12, 14, 16, 18, 20])
print(a)
#sinh ma trận 3 x 5 ngẫu nhiên từ mảng giá trị cho trước
b = np.random.choice([2, 4, 6, 8, 10, 12, 14, 16, 18, 20], size=(3, 5))
print(b)
```

```
4
[[ 2 20 18 14  4]
 [12 16  8  6 18]
 [ 2  2  2 16 14]]
```

3.8. Thay đổi giá trị của ma trận

Các giá trị trong ma trận là bất biến. Để thay đổi giá trị của ma trận ta làm như sau

```
#Tạo ma trận 5 x 5 với số 1
A = np.matrix(np.ones((5, 5)))
print('The matrix\n',A)
np.array(A)[2] = 2
print('Change values of third row with array\n', A)
np.asarray(A)[2] = 2
print('Change values of third row with asarray\n', A)
```

```
The matrix
 [[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
Change values of third row with array
 [[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
Change values of third row with asarray
 [[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [2. 2. 2. 2. 2.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
```

3.9. Sinh dãy số theo khoảng cho trước

3.9.1. Dùng hàm arange

```
#tạo 20 số nguyên từ 1 --> (n-1)
a = np.arange(1, 20)
print(a)
#tạo 20 số nguyên từ 1 --> (n-1), mỗi bước tăng 2 giá trị
b = np.arange(1, 20, 2)
print(b)
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
[ 1  3  5  7  9 11 13 15 17 19]
```

3.9.2. Dùng hàm linspace

```
a = np.linspace(1., 5., num=10)
print(a)
```

```
[1.         1.44444444 1.88888889 2.33333333 2.77777778 3.22222222
 3.66666667 4.11111111 4.55555556 5.        ]
```

3.10. Index và slice

Quy tắc lấy Index trong ma trận

- Giá trị trước dấu ‘,’ là số dòng
- Giá trị sau dấu ‘,’ là số cột
- Muốn chọn một dòng hoặc cột cụ thể thì cần thêm dấu ‘:’

```
#slice
a = np.array([(1, 2, 3), (4, 5, 6)])
print(a)
print('first row:', a[0])
print('second row:', a[1])
#indexing
print('Second column:', a[:,1])
print('first column', a[:, 0])
print('First two values of second row:', a[1, :2])
```

```
[[1 2 3]
 [4 5 6]]
first row: [1 2 3]
second row: [4 5 6]
Second column: [2 5]
first column [1 4]
First two values of second row: [4 5]
```

4. GIỚI THIỆU PANDAS

- Là thư viện mở của Python
- Xử lý dữ liệu dạng bảng
- Pandas dựa trên tầng cơ sở Numpy
- Pandas xử lý dữ liệu chuỗi thời gian
- Dễ dàng xử lý missing data
- Dùng đối tượng Series xử lý data 1-chiều và DataFrame để xử lý data 2-chiều

5. CÀI ĐẶT PANDAS

```
pip install pandas
```

6. TÌM HIỂU ĐỐI TƯỢNG SERIES

Series là cấu trúc dữ liệu 1-chiều.

```
#Tạo pandas series từ list
a = pd.Series([1., 2., 3.])
print(a)
print('First value:', a[0])
b = pd.Series([1., 2., 3.], index=['a', 'b', 'c'])
print(b)
print('First value:', b['a'])
#Tạo pandas series từ dictionary
calories = {"day1": 420, "day2": 380, "day3": 390}
myvar = pd.Series(calories)
print(myvar)
```

```
0    1.0
1    2.0
2    3.0
dtype: float64
First value: 1.0
a    1.0
b    2.0
c    3.0
dtype: float64
First value: 1.0
day1    420
day2    380
day3    390
dtype: int64
```

7. TẠO ĐỐI TƯỢNG DATAFRAME

DataFrame là mảng 2-chiều có nhãn ở 2 trục

Chuyển đổi từ mảng dạng Numpy sang Pandas và ngược lại

```
#Numpy --> Pandas
h = np.array([[1, 2], [3, 4]])
df_h = pd.DataFrame(h)
print('DataFrame:\n', df_h)

#Pandas --> Numpy
numpy_h = np.array(df_h)
print('Numpy array:\n', numpy_h)
```

```
DataFrame:
    0  1
0  1  2
1  3  4
Numpy array:
 [[1 2]
 [3 4]]
```

Tạo DataFrame mảng 2-chiều có tên cột

```
dic = {'Name':['John', 'Smith'], 'Age':[30, 40]}
df_dic = pd.DataFrame(dic)
print(df_dic)
```

```
    Name  Age
0   John   30
1  Smith   40
```

```
#tạo mảng 20 x 4
r = np.random.randn(20, 4)
df = pd.DataFrame(r, columns=['so1', 'so2', 'so3', 'so4'])
print(df.head())  #in 5 dòng đầu tiên
```

```
        so1       so2       so3       so4
0  0.178858  0.218294  0.366942  0.770592
1  0.731135  0.708960  2.175899 -1.195981
2 -1.629464 -0.065397 -1.408343 -1.072836
3  0.471061  0.497364 -1.481091  0.628187
4  0.031584  0.153133  2.494551  0.368162
```

Slice DataFrame

```
#tạo mảng 20 x 4
r = np.random.randn(20, 4)
df = pd.DataFrame(r, columns=['so1', 'so2', 'so3', 'so4'])
print(df.head())  #in 5 dòng đầu tiên
#chọn 1 cột
print('Chọn cột so1\n',df['so1'])
#chọn nhiều dòng
print('Chọn nhiều dòng\n',df[0:3])
#chọn nhiều cột
print('Chọn nhiều cột\n', df[['so1', 'so3']])
#chọn nhiều dòng, nhiều cột
print('Chọn nhiều dòng, nhiều cột\n', df.iloc[:3, :2])
#Xóa cột
df = df.drop(columns=['so1', 'so4'])
print(df)
```

7.1. Nối nhiều DataFrame

```
#Nối 2 DataFrame
df1 = pd.DataFrame({'Name':['John', 'Smith', 'Paul'],
                    'Age':[25, 30, 50]
                    }, index=[0, 1, 2])
df2 = pd.DataFrame({'Name':['Adam', 'Smith'],
                    'Age':[26, 11]
                    }, index=[3, 4])
df_concat = pd.concat([df1, df2])
print('Concate two DataFrame\n', df_concat)
#Xóa những giá trị ở cột Name trùng nhau
print('Drop duplicatevalues in DataFrame\n', df_concat.drop_duplicates('Name'))
#Sắp xếp theo cột
print('Sort values in DataFrame\n',df_concat.sort_values('Age'))
#Đổi tên cột
df_rename = df_concat.rename(columns={'Name':'Surname', 'Age':'Age_pupil'})
print('Rename columns of DataFrame\n', df_rename)
```

```
Concate two DataFrame
     Name  Age
0   John   25
1  Smith   30
2   Paul   50
3   Adam   26
4  Smith   11
Drop duplicatevalues in DataFrame
     Name  Age
0   John   25
1  Smith   30
2   Paul   50
3   Adam   26
Sort values in DataFrame
     Name  Age
4  Smith   11
0   John   25
3   Adam   26
1  Smith   30
2   Paul   50
Rename columns of DataFrame
   Surname  Age_pupil
0    John         25
1   Smith         30
2    Paul         50
3    Adam         26
4   Smith         11
```

7.2. Đọc dữ liệu từ CSV

Ta dùng sau để đọc dữ liệu từ CSV

```
pandas.read_csv(filepath_or_buffer,sep=',
',`names=None`,`index_col=None`,`skipinitialspace=False`)
```

Trong đó:

- filepath_or_buffer: Path or URL with the data
- sep=', ': Define the delimiter to use
- 'names=None': Name the columns. If the dataset has ten columns, you need to pass ten names
- 'index_col=None': If yes, the first column is used as a row index
- 'skipinitialspace=False': Skip spaces after delimiter

```
#Đọc dữ liệu từ CSV
path = '/content/sample_data/Salary_Data.csv'
COLUMNS = ['YearsExperience', 'Salary']
df_csv = pd.read_csv(path, skipinitialspace=True, names=COLUMNS, index_col=False)
print(df_csv.head())
print(df_csv.shape)
df_csv1 = pd.read_csv(path, skipinitialspace=True, index_col=False)
print(df_csv1.head())
print(df_csv1.shape)
```

```
   YearsExperience    Salary
0  YearsExperience    Salary
1              1.1  39343.00
2              1.3  46205.00
3              1.5  37731.00
4              2.0  43525.00
(31, 2)
   YearsExperience   Salary
0              1.1  39343.0
1              1.3  46205.0
2              1.5  37731.0
3              2.0  43525.0
4              2.2  39891.0
(30, 2)
```

7.3. Gom nhóm dữ liệu DataFrame

Một số hàm tính toán theo nhóm như sau

- count
- min
- max
- mean
- median
- standard deviation

Để nhóm dữ liệu theo cột ta dùng hàm **groupby() kết hợp hàm tính toán ở trên**

```
#Gom nhóm dữ liệu trong DataFrame
path = '/content/sample_data/Fish.csv'
df_csv = pd.read_csv(path, skipinitialspace=True, index_col=False)
print(df_csv.head())
print(df_csv.groupby(['Species']).mean().head())
print(df_csv.groupby(['Species'])['Weight'].min().head())
print(df_csv.groupby(['Species', 'Height'])['Weight'].min().head())
```

```
  Species  Weight  Length1  Length2  Length3   Height   Width
0   Bream   242.0     23.2     25.4     30.0  11.5200  4.0200
1   Bream   290.0     24.0     26.3     31.2  12.4800  4.3056
2   Bream   340.0     23.9     26.5     31.1  12.3778  4.6961
3   Bream   363.0     26.3     29.0     33.5  12.7300  4.4555
4   Bream   430.0     26.5     29.0     34.0  12.4440  5.1340
              Weight    Length1    Length2    Length3     Height     Width
Species
Bream     617.828571  30.305714  33.108571  38.354286  15.183211  5.427614
Parkki    154.818182  18.727273  20.345455  22.790909   8.962427  3.220736
Perch     382.239286  25.735714  27.892857  29.571429   7.861870  4.745723
Pike      718.705882  42.476471  45.482353  48.717647   7.713771  5.086382
Roach     152.050000  20.645000  22.275000  24.970000   6.694795  3.657850
Species
Bream     242.0
Parkki     55.0
Perch       5.9
Pike      200.0
Roach       0.0
Name: Weight, dtype: float64
Species  Height
Bream    11.5200    242.0
         12.3778    340.0
         12.4440    430.0
         12.4800    290.0
         12.6700    390.0
Name: Weight, dtype: float64
```

8. BÀI TẬP

1. Tạo mảng 20 số nguyên ngẫu nhiên kích thước 3 x 3 có giá trị trong khoảng [0, 20]. Viết chương trình kiểm tra tất cả các phần tử trong mảng khác 0 (tìm hiểu hàm all của numpy)
2. Tạo mảng 20 số nguyên ngẫu nhiên kích thước 3 x 3 có giá trị trong khoảng [0, 20]. Viết chương trình kiểm tra tồn tại một phần tử trong mảng khác 0 (tìm hiểu hàm any của numpy)
3. Sinh viên tìm hiểu và viết chương trình sử dụng các hàm sau numpy.greater, numpy.greater_equal, numpy.less, và numpy.less_equal để so sánh 2 mảng 1 chiều, 2 chiều.
4. Viết chương trình tạo mảng gồm 10 số 1, 10 số 0 và 10 số 5
5. Viết chương trình tạo mảng số nguyên chẵn từ [30, 70]
6. Viết chương trình tạo ma trận đơn vị 3 x 3 (đường chéo chính bằng 1, các phần tử còn lại 0) (Tìm hiểu hàm identity của numpy)

7. Viết chương trình tạo mảng 10 phần tử trong khoảng [15, 55]. In các giá trị ngoài trừ phần tử đầu và cuối
8. Viết chương trình tạo mảng 20 phần tử trong khoảng [0, 20] và đổi dấu những số nằm trong khoảng [9, 15]
9. Viết chương trình tạo ma trận 3 x 4 và mỗi phần tử có giá trị [10, 21]
10. Viết chương trình tạo ma trận 10 x 10, các phần tử ở biên có giá trị 1, các phần tử bên trong có giá trị 0
11. Viết chương trình tạo ma trận 5 x 5, các phần tử ở đường chéo chính là 1, 2, 3, 4, 5. (tìm hiểu hàm diag của numpy)
12. Viết chương trình tạo mảng 3 x 3 x 3 và tính tổng theo mỗi dòng, mỗi cột. (tìm hiểu hàm sum của numpy)
13. Tạo 2 vector ngẫu nhiên 10 phần tử. Thực hiện inner product (np.dot)
14. Tạo ma trận A= 4 x 3 và vector y gồm 3 phần tử ngẫu nhiên. Thêm vector y vào từng dòng của ma trận A
15. Viết chương trình Pandas để cộng, trừ, nhân, chia 2 Series : [2, 4, 6, 8, 10], [1, 3, 5, 7, 10]
16. Viết chương trình Pandas chuyển đổi cột đầu tiên thành Series (Tìm hiểu thuộc tính ix của pandas)

```
   col1  col2  col3
0     1     4     7
1     2     5     5
2     3     6     8
3     4     9    12
4     7     5     1
5    11     0    11

1st column as a Series:
0     1
1     2
2     3
3     4
4     7
5    11
```

17. Viết chương trình Pandas sắp xếp theo cột của DataFrame 4 x 3 ngẫu nhiên
18. Viết chương trình Pandas thay đổi index của một Series (dùng pandas.reindex)

```
Original Data Series:
A    1
B    2
C    3
D    4
E    5
dtype: int64
Data Series after changing the order of index:
B    2
A    1
C    3
D    4
E    5
dtype: int64
```

19. Viết chương trình Pandas tìm những phần tử trong Series x mà không có trong Series y (Tìm hiểu hàm isin của pandas)

    sr1:
    0 1
    1 2
    2 3
    3 4
    4 5
    sr2:
    0 2
    1 4
    2 6
    3 8
    4 10

    Items of sr1 not present in sr2:
    0 1
    2 3
    4 5

20. Tìm hiểu phép toán numpy.union1d, numpy.intersect1d (hội, giao) và viết chương trình minh họa 2 hàm này trên 2 Series x, y ngẫu nhiên có 20 phần tử
21. Viết chương trình Pandas thêm series y vào series x theo chiều dọc (vertical), chiều ngang (horizontal) (Tìm hiểu hàm append và concat của pandas)

    Series x
    0 0

1 1
2 2
....
7 7
8 8
9 9

Series y
0 p
1 q
2 r
....
7 w
8 x
9 y
dtype: object
Thêm series y vào series x theo chiều dọc (vertical) và chiều ngang (horizontal):
0 1
0 0 p
1 1 q
2 2 r
.....
8 8 x
9 9 y

22. Viết chương trình Pandas tạo và hiển thị dữ liệu DataFrame sau
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
23. Từ bài 22, cho biết 3 dòng đầu tiên
24. Từ bài 22, chọn cột ‘name’ và ‘score’
25. Từ bài 22, chọn những dòng có attemps > 2
26. Từ bài 22, cho biết số dòng, số cột của DataFrame
27. Từ bài 22, cho biết những dòng có score là NaN (tìm hiểu hàm isnull của pandas)
28. Từ bài 22, cho biết những dòng có score trong khoảng (15, 20) (tìm hiểu hàm between của pandas)
29. Từ bài 22, cho biết những dòng có attempt > 2 và score trong khoảng (15, 20)
30. Từ bài 22, cho biết những thay đổi giá trị dòng ‘d’, cột score là 19 (tìm hiểu hàm loc của pandas)