Numpy와 Pandas 데이터 분석

김지선

Contents

1. 넘파이(Numpy)란?

- 다차원 배열/행렬 다루기 좋음
- Python **선형대수 라이브러리**
- C기반 이어서 연산 **속도가 빠름**

  -> import numpy as np

1.2 NumPy 기본 자료형 – ndarray

- 넘파이 기반 데이터 타입
- Python list와 출력형태 같고 속도 빠름
- 객체=np.array([리스트])

```
a=np.array([1,2,3,4])
print(a)
type(a)
```

```
[1 2 3 4]
numpy.ndarray
```

NumPy 타입 확인

- 객체.dtype
  : 객체의 타입(list, ndarray, tuple)
- type(객체)
  : 객체의 데이터 타입(int32, float64)
- 객체.ndim
  : 차원확인
- 객체.shape
  : 구조확인(shape)

```
arr=np.array([[1,2,3],
              [4,5,6]])
print(type(arr)) #전체 타입
print(arr.dtype) #데이터 타입
print(arr.ndim) #차원
print(arr.shape) # shape
```

```
<class 'numpy.ndarray'>
int64
2
(2, 3)
```

NumPy ndarray 편리하게 생성

- np.arange(N)
  - 0~(N-1)까지 연속된 배열 생성

- np.zeros((shape),dtype='자료형')
  - 지정한 shape대로 0으로 채운 ndarray 반환

- np.ones(shape)
  - 지정한 shape대로 1으로 채운 ndarray 반환

NumPy ndarray 편리하게 생성

```
# arange, 10까지 순차적 배열 생성
rangeArr=np.arange(10)
print("range₩n",rangeArr,'₩n')

# zeros, 0으로 채워진 배열 생성
zeroArr=np.zeros((3,2),dtype=int)
print("zeros₩n",zeroArr,'₩n')

# ones, 1로 채워진 배열 생성
oneArr=np.ones((4,3))
print("ones₩n",oneArr)
```

```
range
 [0 1 2 3 4 5 6 7 8 9]

zeros
 [[0 0]
 [0 0]
 [0 0]]

ones
 [[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

NumPy 타입변경 - astype, reshape

- 배열.astype(shape) – 자료형 변경
- 배열.reshape(shape) – 구조(형태) 변경
  ex) 2X5 -> 5X2 , **단 shape 행*열이 같아야함**

```
a=np.array([1,2,3,4])
str_a=a.astype('str')
print(a)
print(str_a)
```

```
[1 2 3 4]
['1' '2' '3' '4']
```

1개, 로우 n개

NumPy reshape 예시

```
arr1=np.arange(15)
print("//original array₩n",arr1)

arr2=arr1.reshape(3,5)
print("₩n//reshape to (3,5) arra

arr3=arr1.reshape(5,3)
print("₩n//reshape to (5,3) arra
```

```
//original array
 [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14]

//reshape to (3,5) array
 [[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]

//reshape to (5,3) array
 [[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]
 [12 13 14]]
```

NumPy reshape에서 -1은?

- N개의 칼럼/행이 되도록 나머지 행/칼럼을 변경 해줌
- 객체.reshape(-1,n): n개의 칼럼을 고정하여 변형
- 객체.reshape(n,-1): n개의 행을 고정하여 변형)

- 예시

(1,10)인 배열->reshape(2,5)=reshape(2,-1)=reshape(-1,5)

NumPy reshape에서 -1 예시

```
a=np.arange(10)
print(a)
a1=a.reshape(2,5)
print('₩n',a1)
a2=a.reshape(2,-1)
print('₩n',a2)
a3=a.reshape(-1,5)
print('₩n',a3)
```

```
[0 1 2 3 4 5 6 7 8 9]

 [[0 1 2 3 4]
 [5 6 7 8 9]]

 [[0 1 2 3 4]
 [5 6 7 8 9]]

 [[0 1 2 3 4]
 [5 6 7 8 9]]
```

NumPy 정렬- sort, argsort

```
arr=np.array([1,5,4,6,3,7,8])
print("원본₩n",arr)
print("₩n정렬후₩n",np.sort(arr))
print("₩n정렬한 인덱스₩n",np.argsort(arr))
```

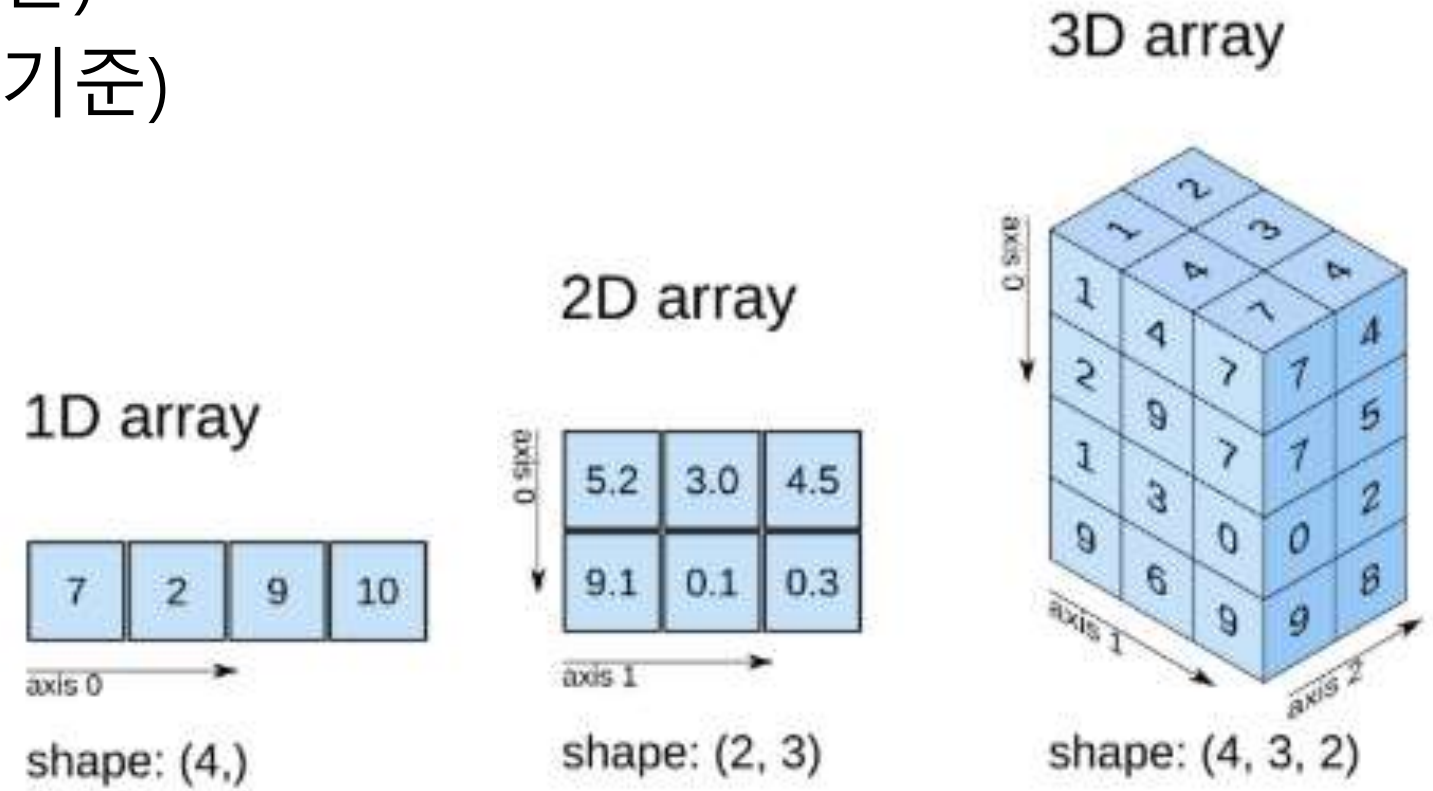
```
원본
 [1 5 4 6 3 7 8]

정렬후
 [1 3 4 5 6 7 8]

정렬한 인덱스
 [0 4 2 1 3 5 6]
```

,원본 유지

래 인덱스(위치) 반환

NumPy 차원(axis=0,1,2)

- axis=0(행 기준, 생략 시 axis=0)
- axis=1(열 기준)
- axis=2(높이 기준)



인덱싱/슬라이싱- 1차원, 2차원,불린

구분	종류	형식
인덱싱	불린 인덱싱	조건 필터링+검색
	1차원	객체[a]
	2차원	객체[a,b]
슬라이싱	1차원	객체[a:b]
	2차원	객체[a:b,c:d]

+인자로 범위 대신 리스트 값 주기 가능(ex[1,4])

NumPy 불린 인덱싱

- 조건 중 참이 되는 것만 출력
  1. 모든 값마다 조건에 대한 True/False값 구하고
  2. True 값만 리턴

```
ndarr=np.arange(7)
under5=ndarr[ndarr<5]
print(ndarr)
print(ndarr<5)
print(under5)
```

```
[0 1 2 3 4 5 6]
[ True  True  True  True  True False False]
[0 1 2 3 4]
```

인덱싱/슬라이싱- 1차원, 2차원,불린

array2d[0:2, 0:2]

1	2	3
4	5	6
7	8	9

array2d[1:3, 0:3]

1	2	3
4	5	6
7	8	9

array2d[1:3, :]

1	2	3
4	5	6
7	8	9

array2d[:, :]

1	2	3
4	5	6
7	8	9

array2d[:2, 1:]

1	2	3
4	5	6
7	8	9

array2d[:2, 0]

1	2	3
4	5	6
7	8	9

출처-https://velog.io/@bangsy/NumPy2%EC%9D%B8%EB%8D%B1%EC%8B%B1

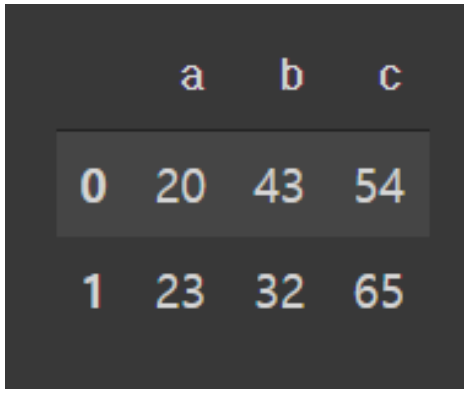
Numpy 총 정리

- **객체=np.array([리스트]) #생성**
- type(객체) ---------- #타입 확인
- 객체.dtype ----------#자료형 확인
- 객체.ndim ---------- #차원 확인
- **객체.shape ---------- #형태 확인**
- np.arange(N) ---------- #0~(N-1)까지 순차적 리스트 확인
- np.zeros((shape).dtype='자료형') #0으로 채운 ndarray 생성
- np.ones(shape) ---------- #1로 채운 ndarray 생성
- 배열.astype(shape) ---------- #자료형 변경
- **배열.reshape(shape) ---------- # 구조 변경**
- Indexing/slicing 방법 ---------- #데이터 출력방법

Pandas의 데이터 타입 – Series, DataFrame

- Series – 칼럼이 1개인 구조체
- DataFrame – 칼럼이 n개인 구조체

(둘다 인덱스 제외)



	a	b	c
0	20	43	54
1	23	32	65

- Pandas 인덱스
  - 최대한 인덱스보다 **칼럼에 뜻**을(이름,나이) **주는게 좋음**
  - 칼럼이 접근/수정 용이하기 때문
    - ⇨ 인덱스는 행 식별의 역할을 함
    - ⇨ 모든 Series와 DataFrame은 칼럼이 없는 인덱스 가짐
    - ⇨ 인덱스는 한 번 생성시 변경 불가능

Pandas Series 생성 방법

- Pd.Series(data)

  ⇨ Data에는 리스트, 튜플, 딕셔너리가 올 수 있음

```
# 리스트로 시리즈 생성
list_data=[20,30,40,50]
a=pd.Series(list_data)
a
```

```
0    20
1    30
2    40
3    50
dtype: int64
```

```
# 튜플로 시리즈 생성
tuple_data=(20,30,40,50)
tuple_Series=pd.Series(tuple_data)
tuple_Series
```

```
0    20
1    30
2    40
3    50
dtype: int64
```

```
# 딕셔너리로 시리즈 생성
dict_data = {'a':1,'b':2,'c':3}
series_data = pd.Series(dict_data)
series_data
```

```
a    1
b    2
c    3
dtype: int64
```

Pandas DataFrame 특징 및 생성방법

- NumPy, Pandas Series에 비해 압도적으로 많이 사용됨
- Pd.DataFrame(data)
  ⇨ Data는 리스트 딕셔너리가 올 수 있음

```
dict_data = {'a0':[1,2,3],'a1':[4,5,6],'a2':[7,8,9],'a3':[10,11,12],'a4':[13,14,15]}
df = pd.DataFrame(dict_data)
df
```

	a0	a1	a2	a3	a4
0	1	4	7	10	13
1	2	5	8	11	14
2	3	6	9	12	15

```
df = pd.DataFrame([[18,'여','경일고'],
                   [17,'남','양동중']],
                  columns = ['나이','성별','학교'])
df
```

	나이	성별	학교
0	18	여	경일고
1	17	남	양동중

Pandas 데이터 불러오고 확인

역할	코드
객체=pd.read_csv(path)	불러오기
객체.head()	상위 n개 보기
객체.describe()	데이터 분포도 확인
객체.info()	칼럼 별 데이터 타입 확인
객체.shape	객체의 크기 반환(행,열)
객체.value_counts()	데이터별 개수 확인
객체.index	인덱스 리스트 반환
객체.columns	칼럼 리스트 반환
객체.values	칼럼 값 반환

Pandas list, numpy 변환

1. 넘파이로 변환
   ⇨ 객체.to_numpy()

2. 딕셔너리로 변환
   ⇨ 객체.to_dict()

3. 리스트로 변환
   ⇨ 객체.tolist()

Pandas DataFrame 데이터 조회

1. 명칭 기반 조회
   ⇨ 객체.loc[행,열]-칼럼명칭 기반 조회(문자)
   ⇨ 단 loc도 행은 숫자로 조회
   ⇨ 객체[열]은 loc 생략 가능

2. 인덱스 기반 조회
   ⇨ 객체.iloc[행,열]-인덱스 기반 조회(숫자)

3. 불린 인덱싱(조건 값 조회)
   ⇨ 객체[객체['칼럼']==값] ([칼럼리스트])]

Pandas 조건 값 입력

- 조건 값 입력
  ⇨ 객체

```
Ex)test2017.loc[test2017['age']==22,'sum']+=1
```

[test2017]

	age	sum
0	22	2 =>3
1	24	3

Pandas 칼럼추가/삭제

1. 추가

   객체['칼럼 A']=값

   ⇨ '칼럼 A' 존재 시 값 할당

   ⇨ 없으면 '칼럼 A' 생성

2. 삭제

   ⇨ del 객체[칼럼]

   ⇨ 객체.drop('칼럼',axis=1)

Pandas 유용한 메서드

- 객체.isnull() ---------- 결측치 확인
- 객체.mean() ---------- 평균값 출력
- 객체.max() ---------- 최댓값 출력
- 객체.min() ---------- 최솟값 출력
- 객체.sum() ---------- 총 합계 출력
- 객체.replace(a,b) ---------- a ⇨ b로 바꿔줌
- 객체.astype('바꿀 자료형') -------- 자료형 바꿔줌
- 객체.to_numeric ---------- 연산 가능한 형태로 변경

Pandas level up !!

```
In [46]: level2017.mean()
```

```
aq1_2     3.847987
aq2_2     3.742463
aq3_2     3.776407
aq4_2     2.565992
aq5_2     2.929159
aq6_2     3.280308
aq7_2     3.312988
aq8_2     3.418090
aq9_2     3.533418
aq10_2    3.663504
aq11_2    2.847987
aq12_2    3.720852
aq13_2    3.404913
aq14_2    3.322897
aq15_2    3.674678
aq16_2    3.048493
aq17_2    2.428105
aq18_2    2.167405
aq19_2    3.239300
aq20_2    1.879401
aq21_2    1.948134
aq22_2    1.995151
aq23_2    1.756589
aq24_2    3.369914
aq25_2    3.094244
```

```
In [47]: level2017.mean().mean()
```

```
2.9855113826280575
```

다음 시간

- 데이터 분석 실습
- 판다스 내용 추가(결측치)
  ⇨ Matplotlib(시각화), 사이킷런 주요 내용 정리
  ⇨ 머신러닝 이론
  ⇨ 알고리즘 소개

Q&A