Python 실습 IV

- Pandas & Numpy

2021-1 산업컴퓨팅개론

목차

- 1. Pandas basic
 - 데이터 구성, 불러오기
 - 데이터 확인, 추출, 필터링
 - 결측치 처리
 - 자료형 변환
 - 데이터 병합
- 2. Numpy basic
 - Array indexing & slicing
 - Array type
 - Array reshape
 - Array join & split
 - Array 연산

패키지 설치

패키지 설치

\$ pip install pandas # numpy 도 함께 설치됨

■ 터미널에 왼쪽과 같이 입력

Pandas 기초

Pandas

- 데이터 셋을 쉽게 다룰 수 있는 파이썬 라이브러리
- Data analysis, cleaning, exploring, manipulating 을 위한 함수 제공
- List, dictionary, numpy array, csv, excel 등 다양한 형태의 정형 데이터 처리 가능
- Series (1차원 데이터), DataFrame (2차원 데이터)
- 소스코드는 다음을 참고 https://github.com/pandas-dev/pandas

Pandas

■ 실습에 필요한 모듈 import

```
import pandas as pd
import numpy as np
```

데이터 구성, 불러오기 (Series)

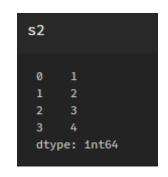
- 1 차원 데이터 처리
- tuple, list, numpy array (1d) 등 1차원 배열을 인풋으로 함.
- Index는 데이터의 각 행을 나타내는 번호. 기본으로 0,1,.. 이지만 index 속성을 통해 지정 가능

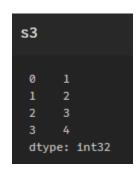
```
# series 구성
s1 = pd.Series( tuple, list or numpy array(1d), index=[0,1,2,..] )
```

```
# Series
tuple_ = (1,2,3,4)
list_ = [1,2,3,4]
array_ld = np.array([1,2,3,4])

s1 = pd.Series(tuple_)
s2 = pd.Series(list_)
s3 = pd.Series(array_ld)
```







데이터 구성, 불러오기 (DataFrame)

- 1, 2 차원 데이터 처리
- 1개 이상의 Series 를 모아 놓은 것.
- tuple, list, numpy array (1d) + list of list, dictionary, numpy array (2d)
- Columns, index 등의 속성으로 컬럼명, 인덱스 지정 가능

■ read_csv (read_excel): csv나 excel 파일 읽기

```
# dataframe 구성 (file 읽기)
df1 = pd.read_csv( file.csv, index_col=0 )
```

데이터 구성, 불러오기 (DataFrame)

- examples
 - Dictionary: key 를 컬럼으로, value 를 리스트 형태의 값으로 지정

```
# DataFrame
# dictionary
mydataset = {
    'student':['mike', 'james', 'dean'],
    'score':[88, 92, 78]
}

df1 = pd.DataFrame(mydataset)
df1

student score

o mike 88
1 james 92
2 dean 78
```

■ Numpy array: array 의 행과 열을 그대로 유지.

데이터 구성, 불러오기 (DataFrame)

- examples
 - read_csv

```
file_path = 'titanic.csv'
df1 = pd.read_csv(file_path)
```

In [18]:	df1.head()	# head(n):	. 전 n .	행 출:	력 # tail((n): D/X/=	₹n # :	출력				
	Passenger Class	Name	Sex	Age	No of Siblings or Spouses on Board	No of Parents or Children on Board	Ticket Number	Passenger Fare	Cabin	Port of Embarkation		Survived
	0 First	Allen, Miss. Elisabeth Walton	Female	29.0	0.0	0.0	24160	211.3	B5	Southampton		Yes
	1 First	Allison, Master. Hudson Trevor	Male	0.9	1.0	2.0	113781	151.6	C22 C26	Southampton	11	Yes
	2 First	Allison, Miss. Helen Loraine	Female	2.0	1.0	2.0	113781	151.6	C22 C26	Southampton	NaN	No
	3 First	Allison, Mr. Hudson Joshua Creighton	Male	30.0	1.0	2.0	113781	151.6	C22 C26	Southampton	NaN	No
	4 First	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	Female	25.0	1.0	2.0	113781	151.6	C22 C26	Southampton	NaN	No

데이터 확인

shape: dataframe 의 행, 열 수를 출력

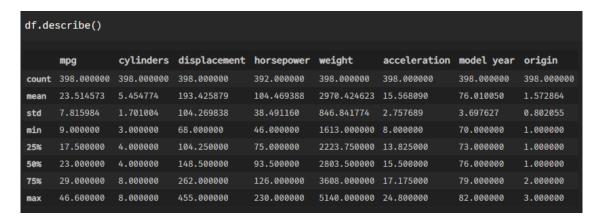


info(): 각 열의 데이터 타입과 전체 데이터프레임
 의 메모리 사용량을 확인 가능

```
dfl.info() # 각 열의 데이터 타입과 전체 데이터프레임의 메모리 사용량을 확인 가능
 <class 'pandas.core.frame.DataFrame'>
 RangeIndex: 1309 entries, 0 to 1308
 Data columns (total 12 columns):
                                     Non-Null Count Dtype
  0 Passenger Class
                                                  object
  4 No of Siblings or Spouses on Board 1309 non-null
  5 No of Parents or Children on Board 1309 non-null
  7 Passenger Fare
                                                  float64
 8 Cabin
  9 Port of Embarkation
                                                  object
  10 Life Boat
  11 Survived
                                    1309 non-null object
 dtypes: float64(4), object(8)
 memory usage: 122.8+ KB
```

columns: 컬럼 확인

■ describe(): 데이터 통계치 확인



■ median(), mean(), std(), max(), min(): 컬럼별 데이터 통계치 출력

```
df1['Age'].mean()
29.88116634799235
```

▪ unique(): 컬럼별 고유값 출력

```
df1['Passenger Class'].unique()
array(['First', 'Second', 'Third'], dtype=object)
```

데이터 추출

■ 특정 행 추출

```
# 특정 행 추출
df.iloc[36] # df.iloc[36, :]
df.loc[36] # df.loc[36, :]
```

```
df1.iloc[36]
 Passenger Class
                                                            First
                                       Bowerman, Miss. Elsie Edith
  Name
  Sex
                                                            Female
                                                              22.0
  Age
 No of Siblings or Spouses on Board
                                                              0.0
  No of Parents or Children on Board
                                                              1.0
 Ticket Number
                                                            113505
 Passenger Fare
                                                              55.0
 Cabin
                                                               E33
 Port of Embarkation
                                                       Southampton
 Life Boat
 Survived
                                                               Yes
 Name: 36, dtype: object
```

■ 특정 열 추출

```
# 특정 열 추출
df['col_name']
```





■ 특정 행 + 특정 열 추출

```
# 특정 행 + 특정 열 추출
df.iloc[36, 2]
df.loc[36, 'col_name']
```

```
df1.iloc[0:2, 3:6]

Age No of Siblings or Spouses on Board No of Parents or Children on Board

0 29.0 0.0

1 0.9 1.0

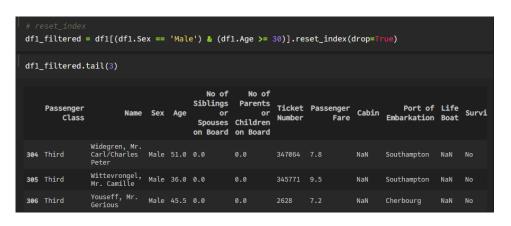
2.0
```

필터링

■ 해당 조건에 만족하는 데이터만 추출

```
# filtering
df[ 조건 ]
```

■ reset_index(): index를 0,1,... 로 재설정. (drop=True로 하면 기존 인덱스는 삭제됨.)



■ ==, >, &, | 사용하여 필터링

```
# 정별이 남자이며, 나이가 30세 이상인 사람만 필터링
df1[(df1['Sex'] == 'Male') & (df1['Age'] >= 30)]
df1[(df1.Sex == 'Male') & (df1.Age >= 30)] # 컬럼에 whitespace 가 없다면 . 으로 접근가능
```

■ isin(): 특정값을 포함하는 데이터만 필터링

```
df1[df1['Port of Embarkation'].isin(['Southampton', 'Cherbourg'])]
```

■ 문자열 메소드를 통한 필터링

```
# 이름에 M이 들어가고 A로 시작하는 인원만 필터링
dfl[dfl.Name.str.contains('M') & dfl.Name.str.startswith('A')]
```

■ columns.difference(): 특정 컬럼을 제외하도록 필터링

```
df1[df1.columns.difference(['Life Boat', 'Ticket Number'])]
```

결측치 처리

- 결측치: 누락된 값. 데이터에서 빈 칸으로 존재하는 값
- isna(): 결측치이면 True, 아니면 False를 반환

dfl.isna()												
	Passenger Class	Name	Sex	Age	No of Siblings or Spouses on Board		Ticket Number	Passenger Fare	Cabin	Port of Embarkation		Survived
0	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	True	False
4	False	False	False	False	False	False	False	False	False	False	True	False

Passenger Class 0
Name 0
Sex 0
Age 263
No of Siblings or Spouses on Board 0
No of Parents or Children on Board 0
Ticket Number 0
Passenger Fare 1
Cabin 1014
Port of Embarkation 2
Life Boat 823
Survived 0
dtype: int64

■ dropna(): 결측치 제거

```
df.dropna(axis=0)
# axis = 0 or 'index' : 결측치 포함 행 제거 (default)
# axis = 1 or 'columns' : 결측치 포함 행 제거
```



결측치 처리

■ fillna(): 결측치 채움

```
df.fillna(0)
# 지정한 값 (0) 으로 채움
df.fillna(method='bfill')# 'backfill', 'ffill', 'pad', default: None
# pad, ffill : 이전 값으로 채움
# backfill, bfill: 다음 값으로 채움
# None: 지정한 값으로 채울 때
```

```
df_example3 = pd.DataFrame({'a':[1,2,3, np.nan ,4], 'b':[5,6,7,8, np.nan]})
df_example3

    a    b
0    1.0    5.0
1    2.0    6.0
2    3.0    7.0
3    NaN    8.0
4    4.0    NaN
```

```
df_example3.fillna(method='bfill')

   a   b
0  1.0  5.0
1  2.0  6.0
2  3.0  7.0
3  4.0  8.0
4  4.0  NaN
```

```
df_example3.fillna(method='ffill')

   a   b
0  1.0  5.0
1  2.0  6.0
2  3.0  7.0
3  3.0  8.0
4  4.0  8.0
```

자료형 변경

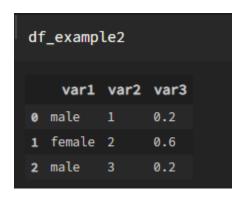
■ astype: 특정 series 의 데이터 타입을 변경

```
df_example = pd.DataFrame({'a':[1, 2, 3],
                              'b':['4', '5', '6'],
                              'c':[7.0, 8.0, 9.0]})
df_example.info()
 <class 'pandas.core.frame.DataFrame'>
 RangeIndex: 3 entries, 0 to 2
 Data columns (total 3 columns):
  # Column Non-Null Count Dtype
            3 non-null
                           1nt64
            3 non-null
                           object
            3 non-null
                           float64
 dtypes: float64(1), int64(1), object(1)
 memory usage: 200.0+ bytes
```

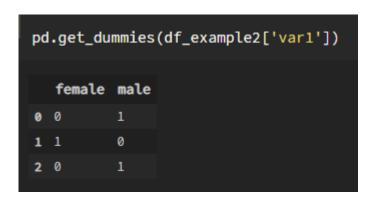
```
df_example['a'] = df_example['a'].astype('float')
df_example['b'] = df_example['b'].astype('int')
df_example['c'] = df_example['c'].astype('str')
df_example.info()
 <class 'pandas.core.frame.DataFrame'>
 RangeIndex: 3 entries, 0 to 2
 Data columns (total 3 columns):
  # Column Non-Null Count Dtype
            3 non-null
                           float64
                          1nt32
            3 non-null
            3 non-null
                          object
 dtypes: float64(1), int32(1), object(1)
 memory usage: 188.0+ bytes
```

자료형 변경 - 더미변수화

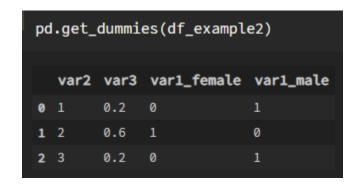
- get_dummies
- Object 타입을 숫자로 변경할 경우 사용
- 'male', 'female' 등 값의 우열이 없는 변수인 경우 1,2 등의 숫자로 바꾸면 정확한 분석이 이뤄질 수 없음.
- One-hot encoding 이란 해당 카테고리에 속하면 1, 그 이외의 카테고리는 0으로 바꾸는 처리방법
- 기존 dataframe



■ Series 를 인풋으로 받으면 각 값을 컬럼으로 하는 one hot encoding 진행



■ DataFrame 을 인풋으로 받으면 prefix를 컬럼명, suffix를 값으로 처리



데이터 병합

• concat(): 두 데이터프레임을 위, 아래로 혹은 양 옆으로 병합

```
# concat
df_concat = pd.concat([df1, df2], axis=0) # axis=0: 위, 아래 (default), # axis=1: 양옆
```

concat

df_concat1 = pd.DataFrame({'a':[1,2,3], 'b':[10, 20, 30], 'c':[100, 200, 300]})

df_concat2 = pd.DataFrame({'a':[4,5,6], 'b':[40, 50, 60], 'c':[400, 500, 600]})

df_concat = pd.concat([df_concat1, df_concat2])

df_concat

a b c

e 1 10 100

df_concat = pd.concat([df_concat1, df_concat2], axis=1)

df_concat

a b c

e 1 10 100

df_concat

a b c a b c

e 1 10 100 4 40 400

1 2 20 200 5 50 500

2 3 30 300 6 60 600

■ axis=0 일 경우 컬럼을 기준으로, 1일 경우 index를 기준으로 병합

```
# concat

df_concat1 = pd.DataFrame({'a':[1,2,3], 'b':[10, 20, 30], 'c':[100, 200, 300]})

df_concat2 = pd.DataFrame({'b':[4,5,6], 'c':[40, 50, 60], 'd':[400, 500, 600]})

df_concat = pd.concat([df_concat1, df_concat2])

df_concat

a b c d

0 1.0 10 10 0 NaN

1 2.0 20 200 NaN

2 3.0 30 300 NaN

0 NaN 4 40 400.0

1 NaN 5 50 500.0

2 NaN 6 60 600.0
```

```
df_concat1 = pd.DataFrame({'a':[1,2,3], 'b':[10, 20, 30], 'c':[100, 200, 300]})
df_concat2 = pd.DataFrame({'a':[4,5,6], 'b':[40, 50, 60], 'c':[400, 500, 600]}, index=[1,2,3])
df_concat = pd.concat([df_concat1, df_concat2], axis=1)
df_concat

    a    b    c    a    b    c

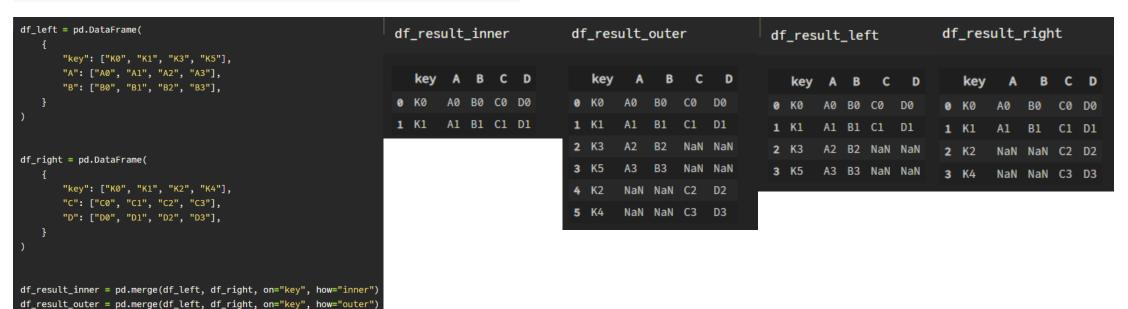
    1.0 10.0 100.0 NaN NaN NaN NaN
1 2.0 20.0 200.0 4.0 40.0 400.0
2 3.0 30.0 300.0 5.0 50.0 500.0
3 NaN NaN NaN NaN 6.0 60.0 600.0
```

데이터 병합

■ merge(): 두 데이터를 키 값을 기준으로 양 옆으로 병합. (Inner, outer, left, right)

```
# merge
df_merge = pd.merge(df1, df2, on='key', how='inner') # 'inner', 'outer', 'left', 'right'
# inner: df1, df2의 키 값의 교집합 병합
# outer: df1, df2의 키 값의 합집합 병합
# left: df1의 키 값을 기준으로 병합
# right: df2의 키 값을 기준으로 병합
```

df_result_left = pd.merge(df_left, df_right, on="key", how="left")
df_result_right = pd.merge(df_left, df_right, on="key", how="right")



데이터 내보내기

to_csv

```
# 파일 내보내기
df.to_csv('내보낼 파일명.csv', index=False)
```

Numpy

- 다차원 배열을 쉽게 처리하고 효율적으로 사용할 수 있도록 지원하는 파이썬 라이브러리
- Numerical Python
- 메모리 내 연속적으로 존재하기 때문에 list 보다 빠른 연산 수행

Array

■ 배열 생성

```
# 1D
np.array([1,2,3,4,5])

# 2D
np.array([[1,2,3], [4,5,6]])

# 3D
np.array([[[1,2,3,4], [5,6,7,8]], [[9,10,11,12], [13,14,15,16]]])
```

Shape, type, ndim 확인

```
arr = np.array([[[1,2,3,4], [5,6,7,8]], [[9,10,11,12], [13,14,15,16]]])

print('shape: ', arr.shape)
print('data type: ', arr.dtype)
print('number of dimensions: ', arr.ndim)

shape: (2, 2, 4)
data type: int32
number of dimensions: 3
```

Array type

dtype

	dtype
정수	Int8, int16, int32, int64
실수	Float16, float32, float64
복소수	Complex64, complex128
그 외	Bool, object, string,

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8], dtype='f8') # $\text{40} + bytes
print(arr.dtype)
print(arr)

float64
[1. 2. 3. 4. 5. 6. 7. 8.]
```

astype

```
print(arr.astype(int).dtype)
int32
```

```
print(arr.astype('i4').dtype)
int32
```

- •i integer
- •b boolean
- •u unsigned integer
- •f float
- •c complex float
- •m timedelta
- •M datetime
- •0 object
- •S string
- •U unicode string
- •V fixed chunk of memory for other type (void)

Array indexing & slicing

- List 를 인덱싱, 슬라이스 하는 방식과 유사
- indexing

```
array([[[ 1, 2, 3, 4], [ 5, 6, 7, 8]], [[ 9, 10, 11, 12], [13, 14, 15, 16]]])
```

```
# 첫 변째 배열의 두 변째 배열에 엑세스
arr[0, 1]
array([5, 6, 7, 8])
```

```
# 마지막 배열의 마지막 배열에 엑세스
arr[-1, -1]
array([13, 14, 15, 16])
```

slice

Array reshape

reshape

```
# reshape
arr = np.array([[1, 2, 3], [4, 5, 6]])
newarr = arr.reshape(-1)
print(arr) # 2D
print()
print(newarr) # 1D flatten

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

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
newarr = arr.reshape(4,3)
print(arr) # 1D
print()
print(newarr) # 2D

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

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

Array Join & split

Concatenate: dimension 추가 없이 두 array 병합

```
arr1 = np.array([[1, 2], [3, 4]])
arr2 = np.array([[5, 6], [7, 8]])

arr_axis0 = np.concatenate((arr1, arr2), axis=0) # axis=0 default
print(arr_axis0)
print()
arr_axis1 = np.concatenate((arr1, arr2), axis=1)
print(arr_axis1)

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

[[1 2 5 6]
    [7 8]]
```

■ Stack: dimension 을 추가하여 두 array 병합

```
arr_stack = np.stack((arr1, arr2))
print(arr_stack)

[[[1 2]
      [3 4]]

[[5 6]
      [7 8]]]
```

■ Array_split: array 를 지정한 개수로 분할하여 리스트로 반환. 다차원 행렬을 분할하는 경우는 axis 설정 가능

```
## array split
arr = np.array([1, 2, 3, 4, 5, 6])
newarr = np.array_split(arr, 3)
print(newarr)

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

Array 연산

■ 기본적인 연산 +, *, -, / (broadcasting 방법): dimension 은 달라도 되지만 shape 이 맞아야 함.

```
# array \(\text{26}\)

arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

print('+: ', arr1 + arr2)

print('*: ', arr1 * arr2)

+: [5 7 9]

*: [4 10 18]
```

```
# diffent dimension (broadcasting)
arr1 = np.array([[1, 2, 3], [1, 2, 3]])
arr2 = np.array([4, 5, 6])

print('+: ', arr1 + arr2)
print('*: ', arr1 * arr2)

+: [[5 7 9]
        [5 7 9]]
        *: [[ 4 10 18]
        [ 4 10 18]]
```

■ dot (matmul): 행렬 곱 연산

```
# np.dot
arr1 = np.array([[1, 2, 3], [1, 2, 3]]) # shape: (2,3)
arr2 = np.array([[4, 5, 6]]).T # shape (3,1), T (transpose)
print(np.dot(arr1, arr2))

[[32]
[32]]
```

과제

과제

- titanic.csv 전처리하기
 - 1. 결측치 제거
 - titanic.csv 를 읽고 각 컬럼 별로 결측치 수가 30% 이상이면 해당 컬럼을 제거하고 30% 이하이면 행을 제거하는 함수를 작성하시오.
 - 함수의 return 에는 전처리된 데이터 프레임이 나와야 함
 - return 할 데이터프레임은 인덱스가 0부터 재설정되어야 함.
 - 2. 자료형 변경
 - 위에서 전처리된 데이터프레임을 읽고 Age 컬럼의 타입을 integer, Sex, Passenger Class, Port of Embarkation 의 경우는 더미변수로 바꾸는 함수를 작성 하시오. (데이터 병합 이용)
 - 함수의 return 에는 전처리된 데이터 프레임이 나와야 함.
 - return 할 데이터프레임은 인덱스가 0부터 재설정되어야 함.