Lecture 6. Pandas 데이터 프레임의 활용(2)

기초 데이터 분석

Recap : Pandas 기본 사용법

- 시리즈와 데이터 프레임의 사용법 : pd.Series(), pd.DataFrame(), columns()
- 데이터 불러오기 / 추출하기 : read_csv() , to_csv()
- 데이터 통계 활용하기 : std(), mean(), describe()
- 데이터 연결과 누락/중복 값 처리 방법 : concat(), join(), fillna(), dropna(), drop_duplicates()

Recap : Pandas 데이터 활용법

- 조건문 활용하여 데이터 조회하기 : **.loc, .iloc()**
- 데이터 프레임 내 문자열 처리 : **str.split(), .slice(), .upper(), .contains()**
- 그룹 연산 : **groupby()**
- 데이터 프레임에 외부 함수 적용하기 : **apply()**
- 시계열 데이터 다루기 : **datetime, dt.date(), dt.time()**

Today : Pandas 활용법 (2)

- 시계열 데이터 다루기 : **datetime, dt.date(), dt.time()**
- 외부 함수 적용하기 : **apply(), assign()**

시계열 데이터 다루기

- **Pandas는 시계열 데이터를 지원**
  - pandas 시계열 자료형 : datetime, time, date 등
  - pd.to_datetime() : 입력받은 string을 datetime으로 변환
    - 변환된 datetime() 객체는 서로 빼거나 더해 timedelta() 단위로 계산이 가능해진다
  - pd.date_range(start, end) : start 부터 end 시간 까지의 시간 범위를 생성

In [3]:
```
date_str = ["2020, 1, 1", "2020, 1, 3", "2020, 1, 5", "2020, 1, 7"]
idx = pd.to_datetime(date_str)
idx
```

```
DatetimeIndex(['2020-01-01', '2020-01-03', '2020-01-05', '2020-01-07'], dtype='datetime64[ns]', freq=None)
```

시계열 데이터 다루기

- **Pandas는 시계열 데이터를 지원**
  - pandas 시계열 자료형 : datetime, time, date 등
  - pd.to_datetime() : 입력받은 string을 datetime으로 변환
    - 변환된 datetime() 객체는 서로 빼거나 더해 timedelta() 단위로 계산이 가능해진다
  - pd.date_range(start, end) : start 부터 end 시간 까지의 시간 범위를 생성

In [3]:
```
print(pd.date_range("2020-1-1", "2020-1-15")) # 1월1일부터 1월15일까지의 시간 범위
print(pd.date_range(start="2020-1-1", periods=15)) # 1월1일부터 15일간의 시간 범위
```

```
DatetimeIndex(['2020-01-01', '2020-01-02', '2020-01-03', '2020-01-04',
               '2020-01-05', '2020-01-06', '2020-01-07', '2020-01-08',
               '2020-01-09', '2020-01-10', '2020-01-11', '2020-01-12',
               '2020-01-13', '2020-01-14', '2020-01-15'],
              dtype='datetime64[ns]', freq='D')
```

데이터 프레임의 함수 연산

- **칼럼들에 연산 적용하기 : apply()**
  - 하나/다수의 칼럼에 복잡한 연산이 필요한 경우 : 자체적으로 함수를 정의
  - 정의된 함수와 연산을 apply(func) 를 통해 적용
  - Arguments
    - 첫 인자는 사전 정의된 작동될 함수의 이름
    - 그 이후 인자는 작동될 함수의 인자를 따름

In [3]:
```
def plus(x):
return x.sepal_length + x.petal_length # iris의 길이를 모두 더함

iris['length_sum'] = iris.apply(plus,axis=1) #인자로 plus 함수를 넣는다.
iris.length_sum.head()
```

```
0    6.5
1    6.3
2    6.0
3    6.1
4    6.4
Name: length_sum,
```

데이터 프레임의 함수 연산

- **새로운 열 생성 : assign()**
  - 이미 존재하는 열을 기반으로 새로운 열을 정의할때 사용
  - assign(새로운_열의_이름 = 수식) 의 형식으로 사용
  - 주로 수식은 람다식을 사용

In [3]:
```
iris.assign(length_sum=lambda x:x.sepal_length + x.petal_length) #인자로 새로운 열 이름과 수식을 제공
iris.length_sum.head()
```

```
0    6.5
1    6.3
2    6.0
3    6.1
4    6.4
Name: length_sum,
```

Today : Pandas를 활용한 실전 데이터 분석

Titanic Dataset

- Kaggle Dataset : 20000 < N (https://www.kaggle.com/c/titanic/)
- 타이타닉호의 승객의 정보를 통해 임의의 사람들의 실종 여부를 맞추는 문제

The Challenge

The sinking of the Titanic is one of the most infamous shipwrecks in history.

On April 15, 1912, during her maiden voyage, the widely considered “unsinkable” RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren’t enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew.

While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others.

In this challenge, we ask you to build a predictive model that answers the question: “what sorts of people were more likely to survive?” using passenger data (ie name, age, gender, socio-economic class, etc).

Today : Pandas를 활용한 실전 데이터 분석

Titanic Dataset

- 총 10개의 컬럼
- 생존 여부, 성별, 나이, 티켓 등급
- 문자열, 숫자 등 다양한 자료형 포함

Data Dictionary

Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
Age	Age in years	
sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Titanic 데이터 불러오기

- **Iris** 데이터와 유사하게 **Seaborn** 라이브러리에서 불러오기

In [3]:
```
import seaborn as sns
titanic = sns.load_dataset('titanic') #titanic 데이터셋 불러오기
titanic.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

Titanic 데이터 불러오기

- 불러온 **titanic** 데이터 **info()** 살펴보기
- 총 **14**개의 열을 가지고 있음
- **int, float, object, category** 등의 자료형
- 모든 열의 **element** 수가 동일하지는 않음

In [3]:
```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

Titanic 데이터 분석 Missions

1. 중복치 처리를 진행해보자
2. 결측치를 처리해보자
3. 각 숫자형 데이터들의 통계치를 구해보자
4. Category 형 데이터에 groupby를 적용해보자
5. String Type 열에도 agg()를 적용해보자
6. Cabin을 여러개의 열로 분할해보자
7. 나만의 경향성을 찾아보자