데이터분석 w/ 파이썬

Pandas 소개

한국폴리텍대학 성남캠퍼스 인공지능소프트웨어과
이혜정 교수

1

Pandas의 개념

- **데이터를 조작하고 분석하는 데 사용하는 파이썬 라이브러리 (파이썬계의 엑셀)**
  - Data Table을 다루기 위한 도구로 가장 적합
- **데이터를 다루기 위해 넘파이(Numpy)를 기본적으로 사용**
  - Numpy : 파이썬에서 배열을 다루는 최적의 라이브러리
  - 넘파이를 효율적으로 사용하기 위해 인덱싱, 연산, 전처리 등 다양한 함수 제공

2

Pandas의 데이터 유형

- 자료구조, 구조화된 데이터 형식, 데이터를 담는 그릇

- 데이터프레임(DataFrame) : 데이터 테이블 전체 객체
- 시리즈(Series) : 각 열 데이터를 다루는 객체

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	weight_0
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.9	4.98	1
1	0.02731	0.0	7.07	0	0.469	6.421	48.9	4.9671	2	242.0	17.8	396.9	9.14	1
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	1
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	1
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.9	5.33	1

시리즈(Series)

데이터프레임 중 하나의 열에 해당하는 데이터의 모음 객체

데이터프레임(DataFrame)

데이터 테이블 전체를 포함하는 객체

3

Pandas의 데이터 유형

시리즈	• 1차원 데이터 • 리스트와 유사하나 리스트와 달리 인덱스를 부여할 수 있음 • sr 별칭을 많이 사용
데이터프레임	• 2차원 데이터 • 각각 인덱스를 가진 행(인덱스로 구분)과 열(열 이름으로 구분)로 이루어진 표 형태로 저장 • df 별칭을 많이 사용

4

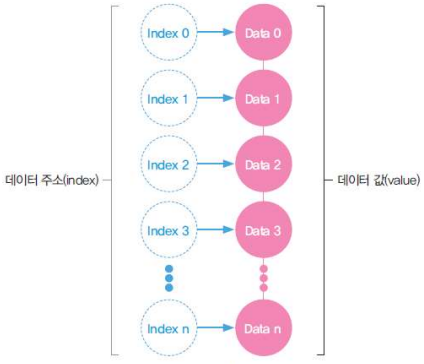
Pandas

Series

5

Series

- **데이터가 순차적으로 나열된 1차원 배열의 형태를 갖음**
  - List와는 달리 index를 가지고 있음
  - list, dict, ndarray(numpy) 등 다양한 데이터 타입이 시리즈 객체 형태로 변환되기도 함
  - 생성된 데이터프레임(DataFrame) 안에 포함될 수 있음

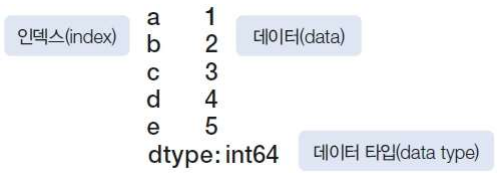


[그림 1-2] 시리즈 구조

6

Series

- **시리즈 객체 생성 시, 세 가지 속성(property) 생성**
  - 데이터(data) : 기존 다른 객체처럼 값을 저장하는 요소
  - 인덱스(index) : 항상 0부터 시작하고, 숫자로만 할당하는 값
    - 시리즈 객체에서는 숫자, 문자열, 0 외의 값으로 시작하는 숫자, 순서가 일정하지 않은 숫자를 입력할 수도 있음
    - 시리즈 객체에서는 인덱스 값의 중복을 허용
  - 데이터 타입(data type) : 넘파이의 데이터 타입과 일치
    - 판다스는 넘파이의 래퍼(wrapper) 라이브러리, 넘파이의 모든 기능 지원하고 데이터 타입도 그대로 적용



실습

7

Series

- **시리즈 객체는 객체의 이름을 변경할 수 있음**
  - 열의 이름을 지정해주는 방식
  - 인덱스 이름도 추가로 지정 가능

```
In [6]: example_obj.name = "number"
        example_obj.index.name = "id"
        example_obj
```

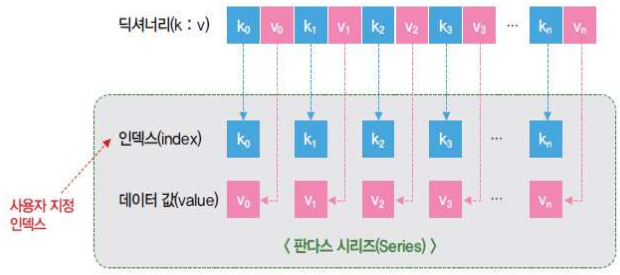
```
Out [6]: id
         a  1
         b  2
         c  3
         d  4
         e  5
         Name: number, dtype: int64
```

8

Series

- **시리즈 객체 생성하기**
  - 데이터프레임 객체를 먼저 생성하고 각 열에서 시리즈 객체를 뽑는 것이 일반적인 방법
  - 다양한 시퀀스형 데이터 타입으로 저장 가능
    - 리스트, 딕셔너리 등

딕셔너리 → 시리즈 변환: `pandas.Series( 딕셔너리 )`



[그림 1-3] 딕셔너리 → 시리즈 변환

9

Series

- **시리즈 객체 생성하기**
  - 데이터프레임 객체를 먼저 생성하고 각 열에서 시리즈 객체를 뽑는 것이 일반적인 방법
  - 다양한 시퀀스형 데이터 타입으로 저장 가능
    - 리스트, 딕셔너리 등
- **딕셔너리를 시리즈로 변환하기**
  - 딕셔너리와 시리즈의 구조가 비슷하기 때문에 딕셔너리를 시리즈로 변환하는 방법 많이 사용
  - 딕셔너리의 키(key) → index, 키에 매칭되는 값(value) → data

```
In [7]: dict_data = {"a":1, "b":2, "c":3, "d":4, "e":5}
        example_obj = Series(dict_data, dtype=np float
        name="example_data")
        example_obj
```

```
Out [7]: a   1.0
         b   2.0
         c   3.0
         d   4.0
         e   5.0
         Name: example_data, dtype: float32
```

10

Series

- **판다스의 모든 객체는 인덱스 값을 기준으로 생성**
  - 기존 데이터에 인덱스 값을 추가하면 NaN 값이 출력됨

In [8]:
```
dict_data_1 = {"a":1, "b":2, "c":3, "d":4, "e":5}
indexes = ["a","b","c","d","e","f","g","h"]
series_obj_1 = Series(dict_data_1, index=indexes)
series_obj_1
```

Out [8]:
```
a   1.0
b   2.0
c   3.0
d   4.0
e   5.0
f   NaN
g   NaN
h   NaN
dtype: float64
```

11

Series

- **투플을 시리즈로 변환하기**
  - 투플도 리스트처럼 딕셔너리의 키에 해당하는 값이 없어서 시리즈로 변환할 때 정수형 위치 인덱스가 자동 지정

In
```
tup_data = ('영인', '2010-05-01', '여', True)
sr = pd.Series(tup_data, index=['이름', '생년월일', '성별', '학생여부'])
sr
```

Out
```
이름              영인
생년월일    2010-05-01
성별               여
학생여부          True
dtype: object
```

12

Series

- **원소 선택**
  - 대괄호 안에 인덱스 입력 : 정수형 위치 인덱스, '인덱스 이름'

In	print(sr[0]) print(sr['이름'])
Out	영인 영인

  - 여러 개 원소 선택 : 인덱스 리스트, 인덱스 범위 지정 [start : end +1]

In	print(sr[1, 2], '\n') print(sr[['생년월일', '성별']])
Out	생년월일 2010-05-01 성별 여 dtype: object 생년월일 2010-05-01 성별 여 dtype: object

In	print(sr[1 : 4], '\n') print(sr['생년월일' : '학생여부'])
Out	생년월일 2010-05-01 성별 여 학생여부 True dtype: object 생년월일 2010-05-01 성별 여 학생여부 True dtype: object

13

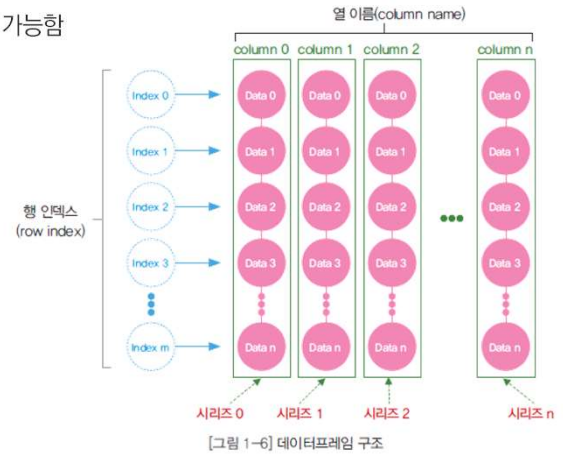
Pandas

DataFrames

14

DataFrames

- **데이터 테이블 전체를 지칭하는 객체**
  - 여러 개의 시리즈(Series) 묶음으로 구성
  - 넘파이 배열의 특성을 그대로 가짐
  - 열별로 다른 데이터 타입이 가능함



[그림 1-6] 데이터프레임 구조

15

DataFrames

- **인덱싱**
  - 열과 행 각각 사용하여 하나의 데이터에 접근

인덱스	열	foo	bar	baz	qux
A	→	0	x	2.7	True
B	→	4	y	6	True
C	→	8	z	10	False
D	→	−12	w	NA	False
E	→	16	a	18	False

그림 4-4 데이터프레임 객체

16

DataFrames 생성

- **데이터프레임을 직접 생성**
  - 딕셔너리 타입 데이터에서 키(key)는 열 이름, 값(value)은 시퀀스형 데이터 타입을 넣어 각 열의 데이터로 만듦

In
```
raw_data = {'first_name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],
'last_name': ['Miller', 'Jacobson', 'Ali', 'Milner', 'Cooze'],
'age': [42, 52, 36, 24, 73],
'city': ['San Francisco', 'Baltimore', 'Miami', 'Douglas', 'Boston']}

df = pd.DataFrame(raw_data, columns = ['first_name', 'last_name', 'age', 'city'])

df
```

17

DataFrames 생성

- **'read_확장자' 함수로 데이터 바로 로딩**
  - .csv나 .xlsx 등 스프레드시트형 확장자 파일에서 데이터 로딩

In
```
# 데이터 URL을 변수 data_url에 넣기
data_url = 'https://archive.ics.uci.edu/ml/machine-learning-
databases/housing/housing.data'
# csv 데이터 로드
df_data = pd.read_csv(data_url, sep='\s+', header = None)

df = pd.DataFrame(df_data)
Df
```

  - Read_csv
    - 정규식 \s+ : 공백(스페이스) 하나 이상으로 열을 구분
    - header=None: 데이터 파일의 첫 번째 행을 열 이름으로 사용하지 않음
      - 데이터 파일에 열 이름이 없을 경우에 이렇게 설정하면 Pandas가 자동으로 열 번호를 열 이름으로 사용.

18

Dataframes의 열 다루기

- 데이터 생성시, 열 이름을 한정하면 해당 열만 추출

```
In  DataFrame(raw_data, columns = ["age", "city"])
```

- 데이터가 존재하지 않는 열을 추가하면 해당 열에는 NaN 값들 추가

```
In  DataFrame(raw_data,
              columns = ["first_name","last_name","age", "city", "debt"]
             )
```

19