빅데이터 처리 및 응용

제 6 강
빅데이터 분석을 위한 파이썬

김 대 경
한양대학교 응용수학과

빅데이터 분석을 위한 파이썬

차 례

- Numpy의 기초
- 데이터 분석을 위한 Plot
- Pandas의 기초
- 데이터 전처리

■ Pandas의 기초

- Pandas는 데이터 처리와 분석을 위한 파이썬 라이브러리로써 R의 data.frame과 유사한 DataFrame이라는 데이터 구조를 기반으로 만들어졌음
- Pandas에는 크게 두 가지 형태의 데이터 형식이 있음
  1) 시리즈(Series) 형태의 1차원 배열 형식
  2) 데이터 프레임(Data Frame) 형태의 2차원 배열 형식

❖ 시리즈 형식

- pandas의 시리즈는 복수의 행(Row)으로 이루어진 하나의 열의 구조를 갖고 있으며 색인(index)을 이용하여 원하는 데이터에 접근할 수 있음

```
import pandas as pd

d = pd.Series([1,2,3,4]); type(d)
```

```
pandas.core.series.Series
```

```
print(d)
```

```
0    1
1    2
2    3
3    4
dtype: int64
```

```
d = pd.Series([9,2,5,4], index=['B','D','A','C']); print(d)
```

```
B    9
D    2
A    5
C    4
dtype: int64
```

```
d['B']
```

```
9
```

```
d[['A','D']]
```

```
A    5
D    2
dtype: int64
```

```
d.index
```

```
Index(['B', 'D', 'A', 'C'], dtype='object')
```

```
print(sorted(d.index))
print(sorted(d.values))
```

```
['A', 'B', 'C', 'D']
[2, 4, 5, 9]
```

```
d.reindex(sorted(d.index))
```

```
A    5
B    9
C    4
D    2
dtype: int64
```

```
x = pd.Series([9,2,5,4], index=['B','D','A','C'])
y = pd.Series([10,20,50,40], index=['B','D','E','F'])
x+y
```

```
A     NaN
B    19.0
C     NaN
D    22.0
E     NaN
F     NaN
dtype: float64
```

❖ 데이터프레임 형식

- 데이터프레임(DataFrame)은 2차원 배열의 개념을 가지고 있으며 행과 열로 구성되어 있음.
- 일반적으로 열에 대한 각각의 이름이 부여됨

```
data = {'age':[20,45,15,50], 'name':['서현','영희','민우','재범'], 'height':[165.5,160,179,172.5]}
data
```

```
{'age': [20, 45, 15, 50],
 'name': ['서현', '영희', '민우', '재범'],
 'height': [165.5, 160, 179, 172.5]}
```

```
x = pd.DataFrame(data); type(x)
```

```
pandas.core.frame.DataFrame
```

```
x=pd.DataFrame(data,columns=['name','age','height']); x
```

	name	age	height
0	서현	20	165.5
1	영희	45	160.0
2	민우	15	179.0
3	재범	50	172.5

```
x.name
```

```
0    서현
1    영희
2    민우
3    재범
Name: name, dtype: object
```

```
x['name']
```

```
0    서현
1    영희
2    민우
3    재범
Name: name, dtype: object
```

```
index = [True, False, True, False]
print(x[index])
```

```
   name  age  height
0  서현   20   165.5
2  민우   15   179.0
```

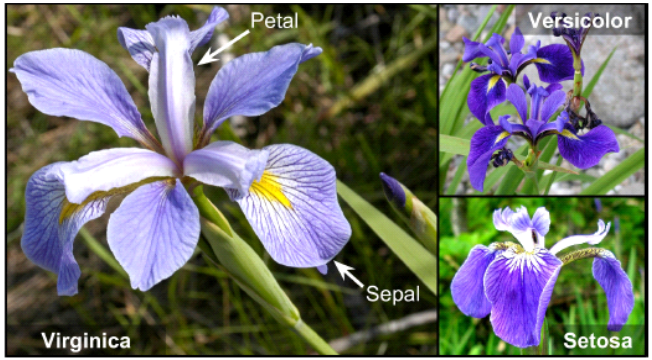
```
print(x.mean(axis=0))
```

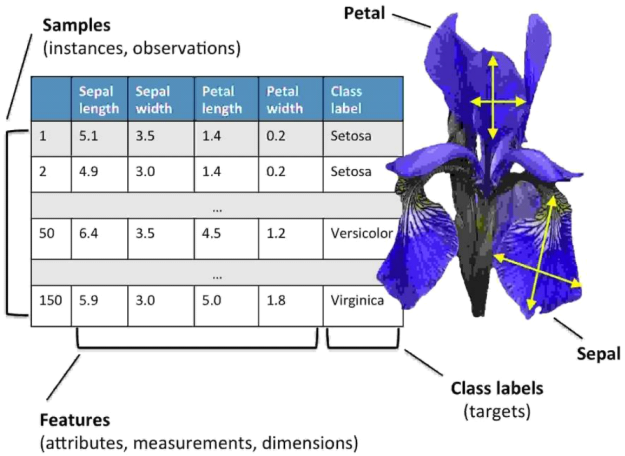
```
age       32.50
height   169.25
dtype: float64
```

- 붓꽃 데이터 예제

붓꽃 데이터셋 가져오기

- ✓ 붓꽃 데이터는 꽃받침(sepal) 길이와 너비, 꽃잎(petal) 길이와 너비 등 네 개의 특성(4차원)을 갖고 있음
- ✓ pandas 라이브러리를 사용하여 UCI 머신 러닝 저장소에서 붓꽃 데이터셋을 DataFrame 객체로 로드(load)하여 가져옴.





	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

```
import pandas as pd

df = pd.read_csv('https://archive.ics.uci.edu/ml/'
        'machine-learning-databases/iris/iris.data', header=None)

df.head()
```

0	1	2	3	4	
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5	3.6	1.4	0.2	Iris-setosa

```
df.tail()
```

0	1	2	3	4	
145	6.7	3	5.2	2.3	Iris-virginica
146	6.3	2.5	5	1.9	Iris-virginica
147	6.5	3	5.2	2	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3	5.1	1.8	Iris-virginica

데이터셋의 재구성

- ✓ 50개의 Iris-setosa와 50개의 Iris-versicolor 꽃에 해당하는 처음 100개의 클래스 레이블을 추출함.
- ✓ 클래스 레이블을 두 개의 정수 클래스 1(versicolor)과 −1(setosa)로 바꾼 다음 벡터 “y”에 저장함.
- ✓ pandas의 DataFrame의 “values” 메서드는 넘파이 배열을 반환함.
- ✓ 마찬가지로 100개의 훈련 샘플에서 첫 번째 특성 열(꽃받침의 길이)과 세 번째 특성 열(꽃잎의 길이)을 추출하여 특성 행렬 “X”에 저장함.

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

# setosa와 versicolor를 선택함
y = df.iloc[0:100, 4].values
y = np.where(y == 'Iris-setosa', -1, 1)

# 꽃받침 길이와 꽃잎 길이를 추출함
X = df.iloc[0:100, [0, 2]].values
```

```
print(y.shape); print(type(y))
```

```
(100,)
<class 'numpy.ndarray'>
```

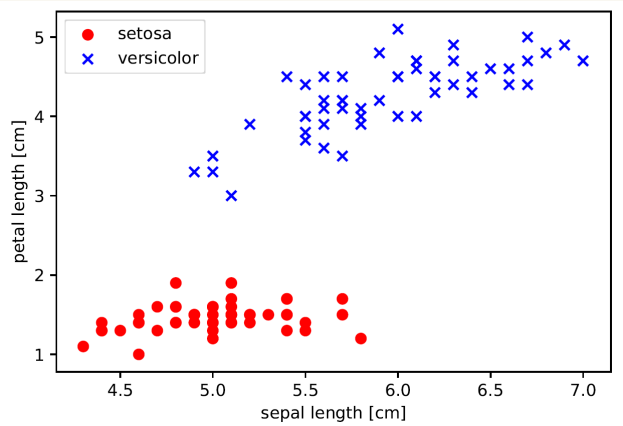
```
print(X.shape); print(type(X))
```

```
(100, 2)
<class 'numpy.ndarray'>
```

```
# 산점도를 그림
plt.scatter(X[:50,0], X[:50,1],
            color='red', marker='o', label='setosa')
plt.scatter(X[50:100,0],X[50:100,1],
            color='blue', marker='x', label='versicolor')

plt.xlabel('sepal length [cm]')
plt.ylabel('petal length [cm]')
plt.legend(loc = 'upper left')

plt.show()
```



■ 데이터 전처리와 데이터 정규화

- 데이터마이닝 또는 머신러닝의 회귀, SVM 등 대부분의 알고리즘은 데이터의 특성(속성) 간 값의 범위(스케일)의 격차가 너무 크면 그 성능이 떨어지거나 개선되지 않는 경향이 있음
- 데이터에 알고리즘을 적용하기 전에 중요한 변환 중 하나는 데이터의 전처리, 즉 특성 스케일링(feature scaling)임

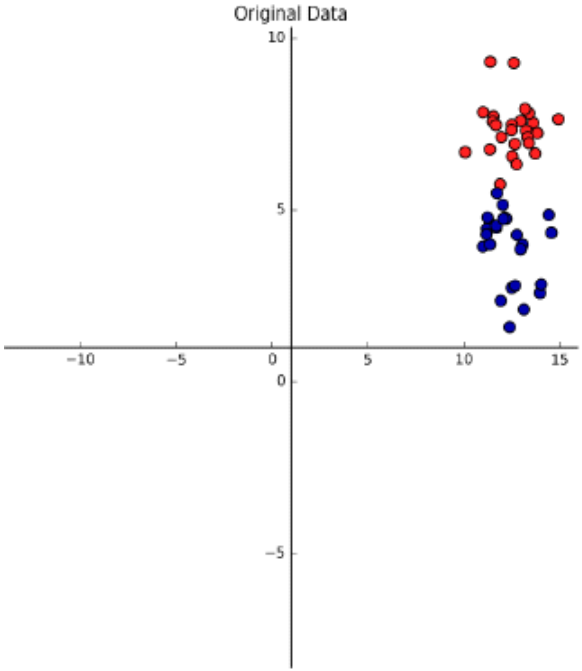
age	income
20	100000
30	20000
40	500000
⋮	⋮

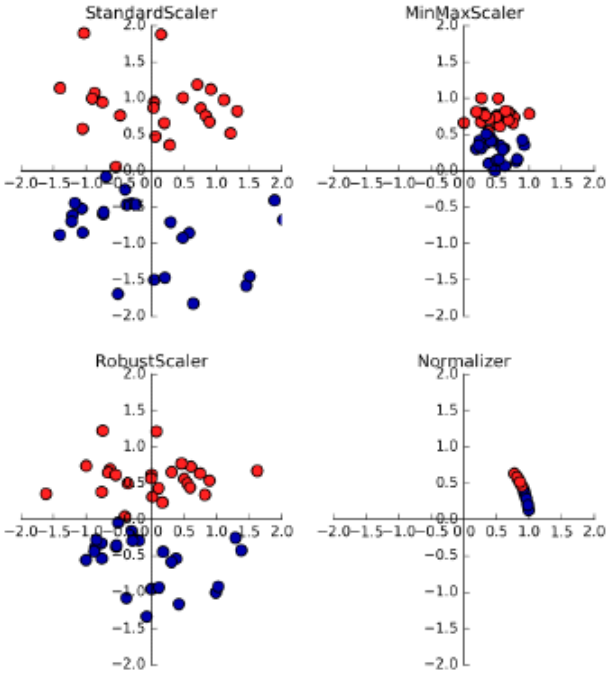
⇨

age	income
0.2	0.2
0.3	0.04
0.4	1
⋮	⋮

❖ 대표적인 전처리 방법

1) StandardScaler (범용적)
   - ✓ 모든 특성의 성분 값의 평균을 0, 분산을 1로 변환함.
2) RobustScaler
   - ✓ 평균과 분산 대신 각 특성의 중앙값, 사분위 값을 사용함.
   - ✓ $\frac{x - q_2}{q_3 - q_1}$
3) MinMaxScaler (범용적)
   - ✓ 모든 특성의 성분 값을 정확하게 0과 1사이에 위치하도록 변환함.
4) Normalizer
   - ✓ 특성 벡터의 유클리디안 길이가 1이 되도록 정규화 함.


Original Data


StandardScaler
MinMaxScaler
RobustScaler
Normalizer