

Data Analysis with Pandas

Pandas 기초 II

Jinwook Seo, Ph.D.

Human-Computer Interaction Lab

Department of Computer Science and Engineering

Seoul National University

지난 강의 내용

- Pandas 소개
- Series
- DataFrame
- 인덱싱

강의 내용

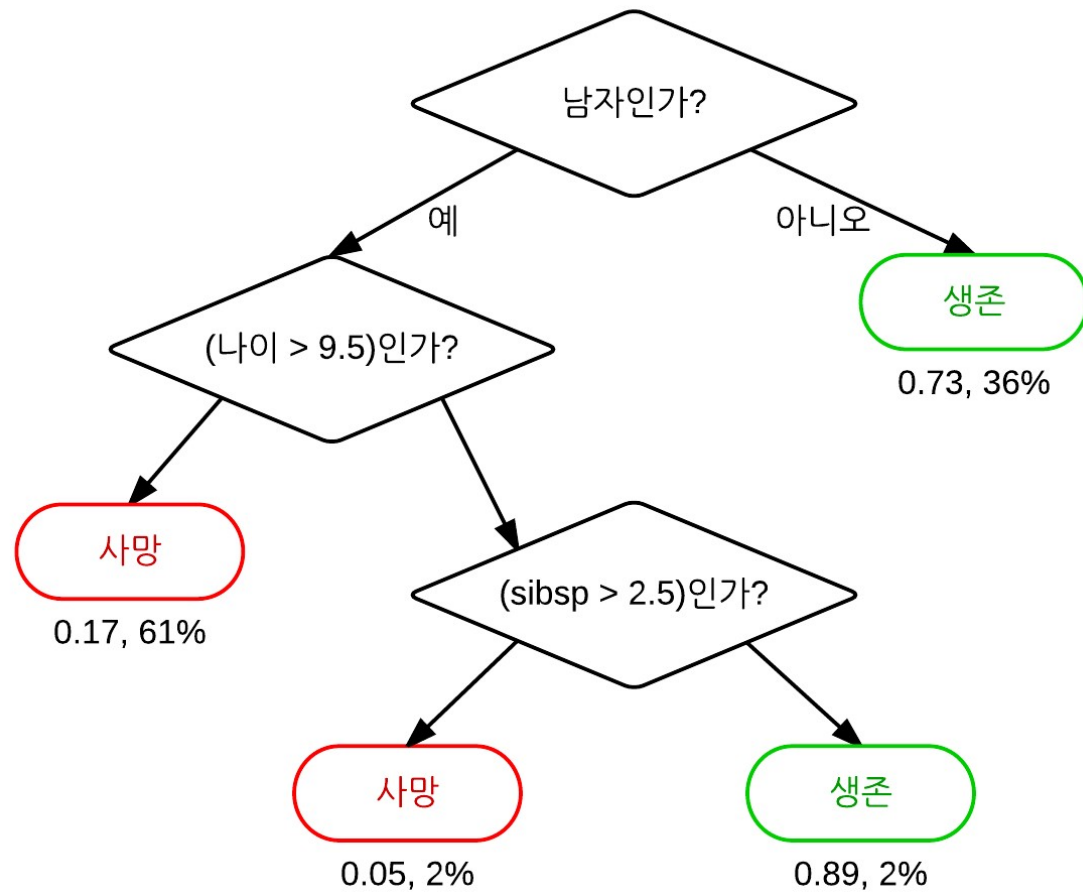
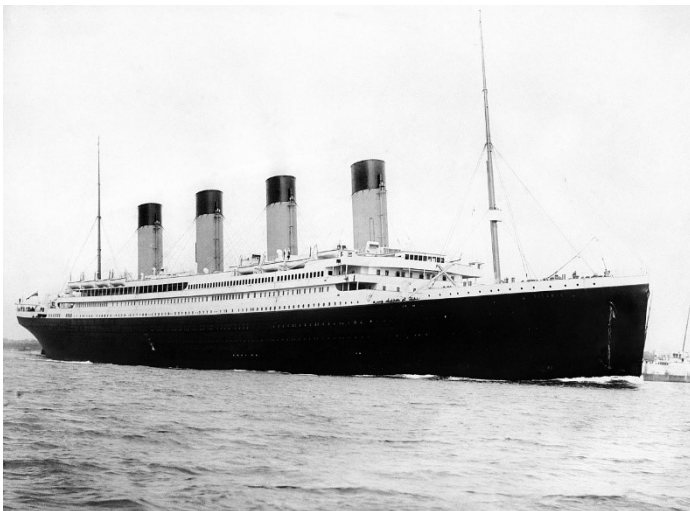
- 기계 학습의 기초
- Pandas를 활용한 결측 데이터 처리
 - 결측 데이터 검색
 - 결측 데이터 제거
 - 결측 데이터 채우기
- Pandas를 이용한 파일 입출력

대표적인 인공지능

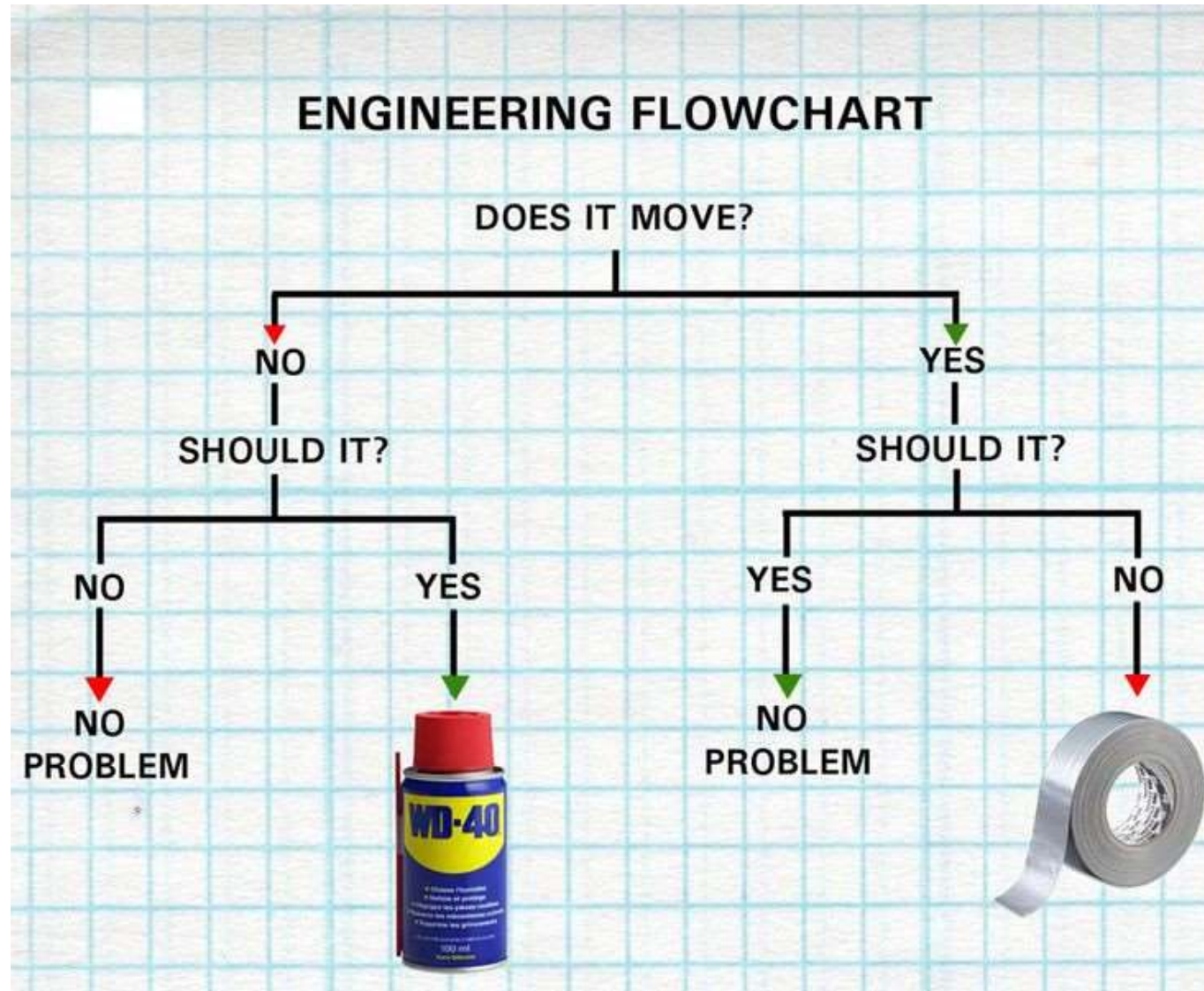


의사 결정 트리

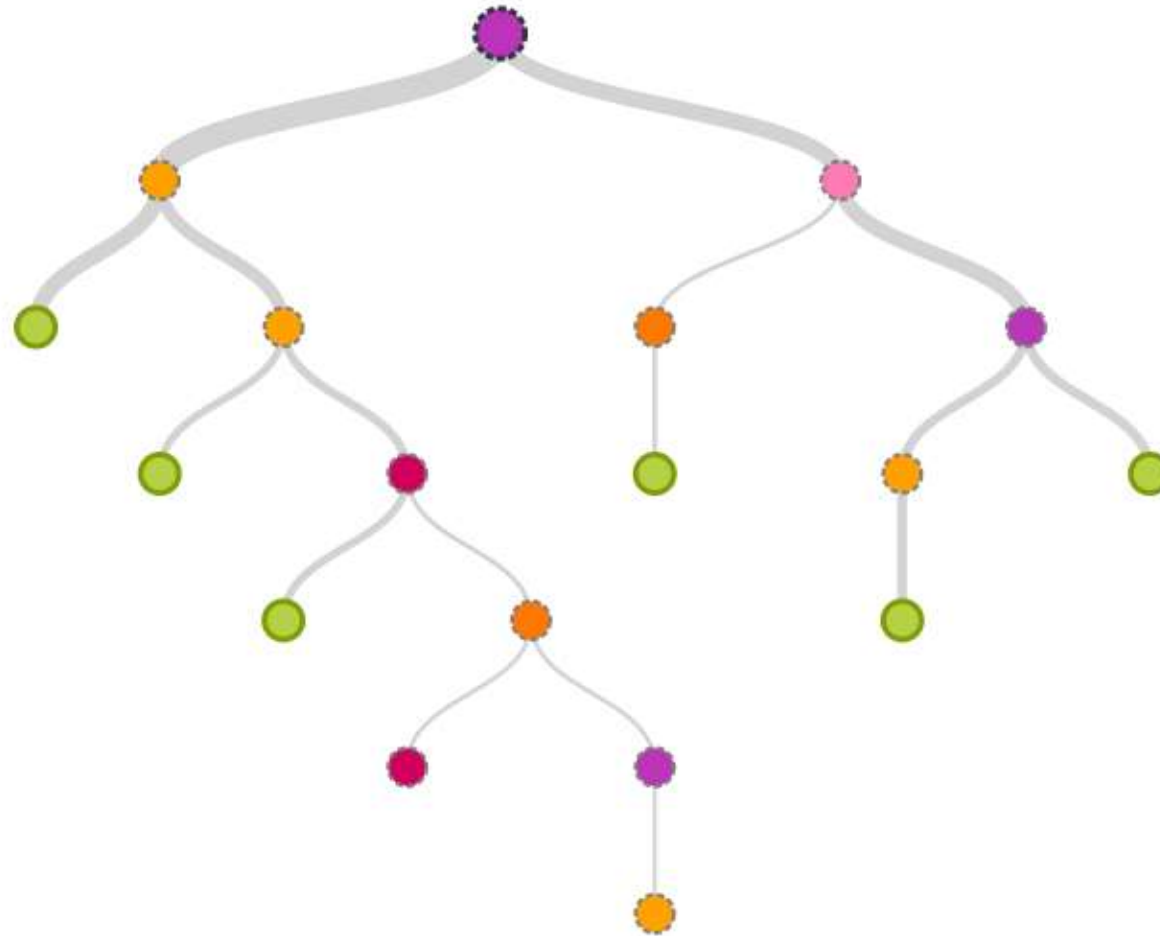
- 타이타닉호 탑승객의 생존 여부



의사 결정 트리

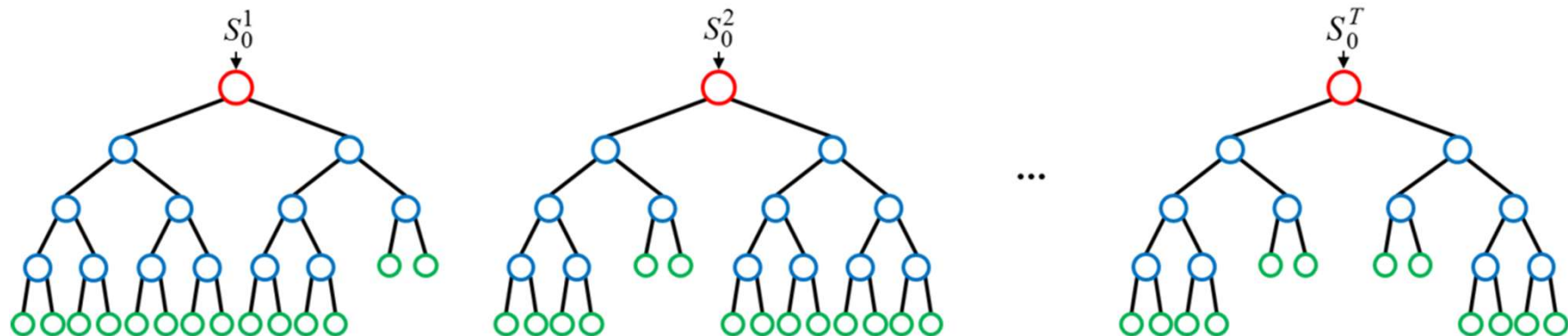


의사 결정 트리의 모양



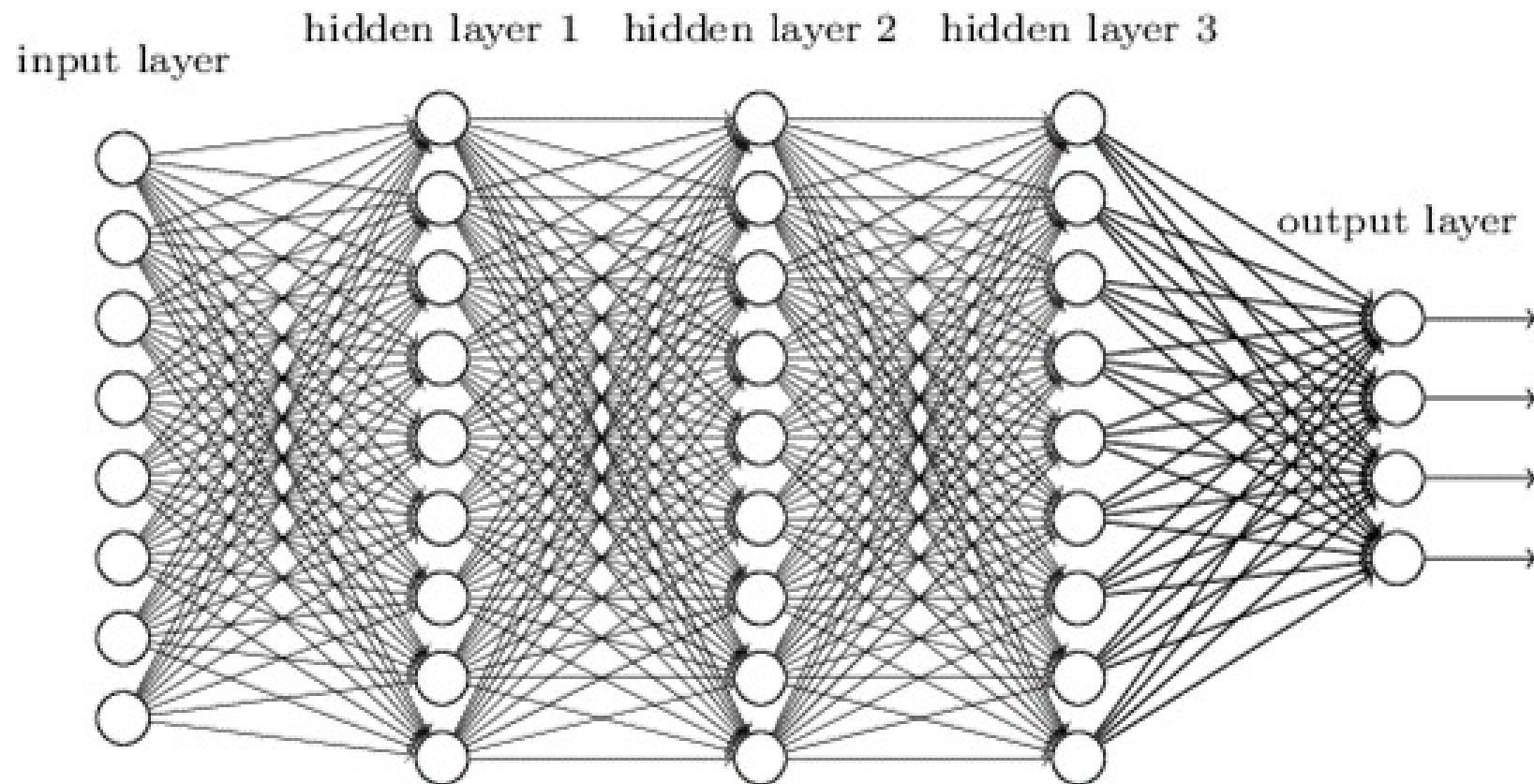
랜덤 포레스트

- 임의성에 의해 서로 조금씩 다른 특성을 갖는 트리들로 구성
- 다수의 의사 결정 트리로부터 값을 받아와서 최종 결과를 도출



딥러닝 (Deep learning)

Deep neural network



기계 학습

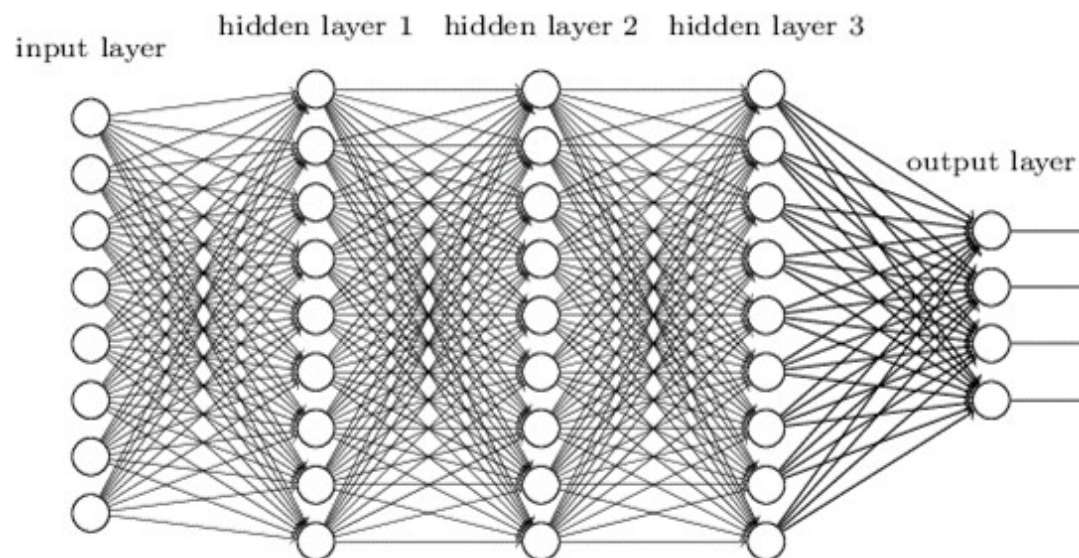
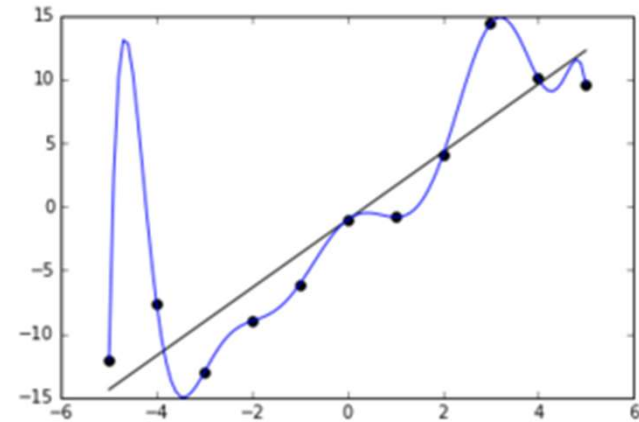
기계학습
(Machine Learning)

인공신경망
(Neural Network)

딥 러닝
(Deep Learning)

딥러닝의 2가지 난제

- Overfitting
- Vanishing gradient



기계 학습의 분류

- 지도학습 (supervised learning)
 - 어떤 입력에 대해서 어떤 결과가 나와야 하는지 사전 지식을 갖고 있는 경우
- 비지도학습 혹은 자율학습 (unsupervised learning)
 - 입력은 있지만 정해진 출력이 없는 경우
- 강화학습 (reinforcement learning)
 - 자신과 환경의 상호관계에 따라 행동을 개선해 나가는 방법

지도학습 (supervised learning)



비지도학습 (unsupervised learning)



airplane

automobile

bird

cat

deer

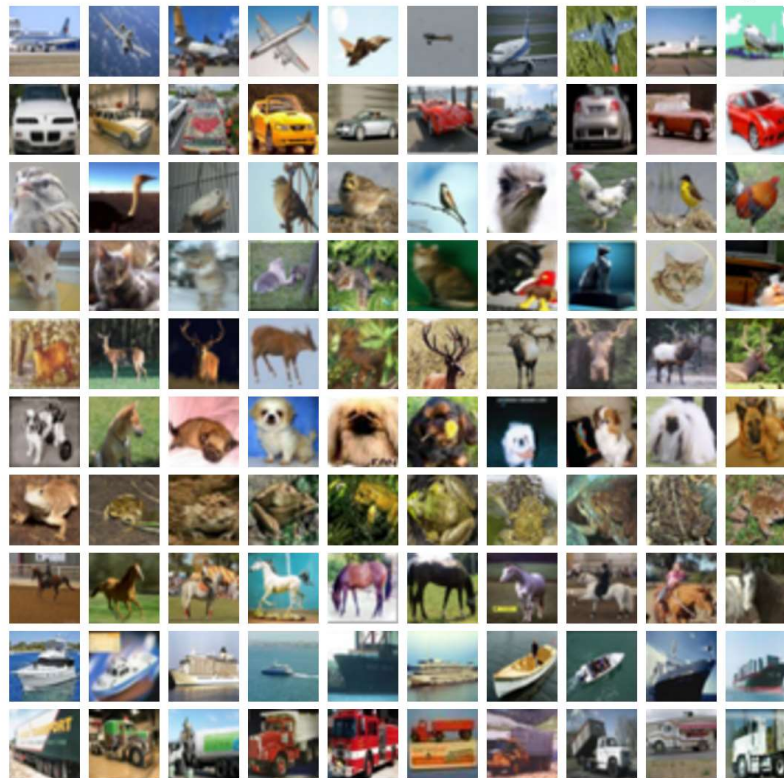
dog

frog

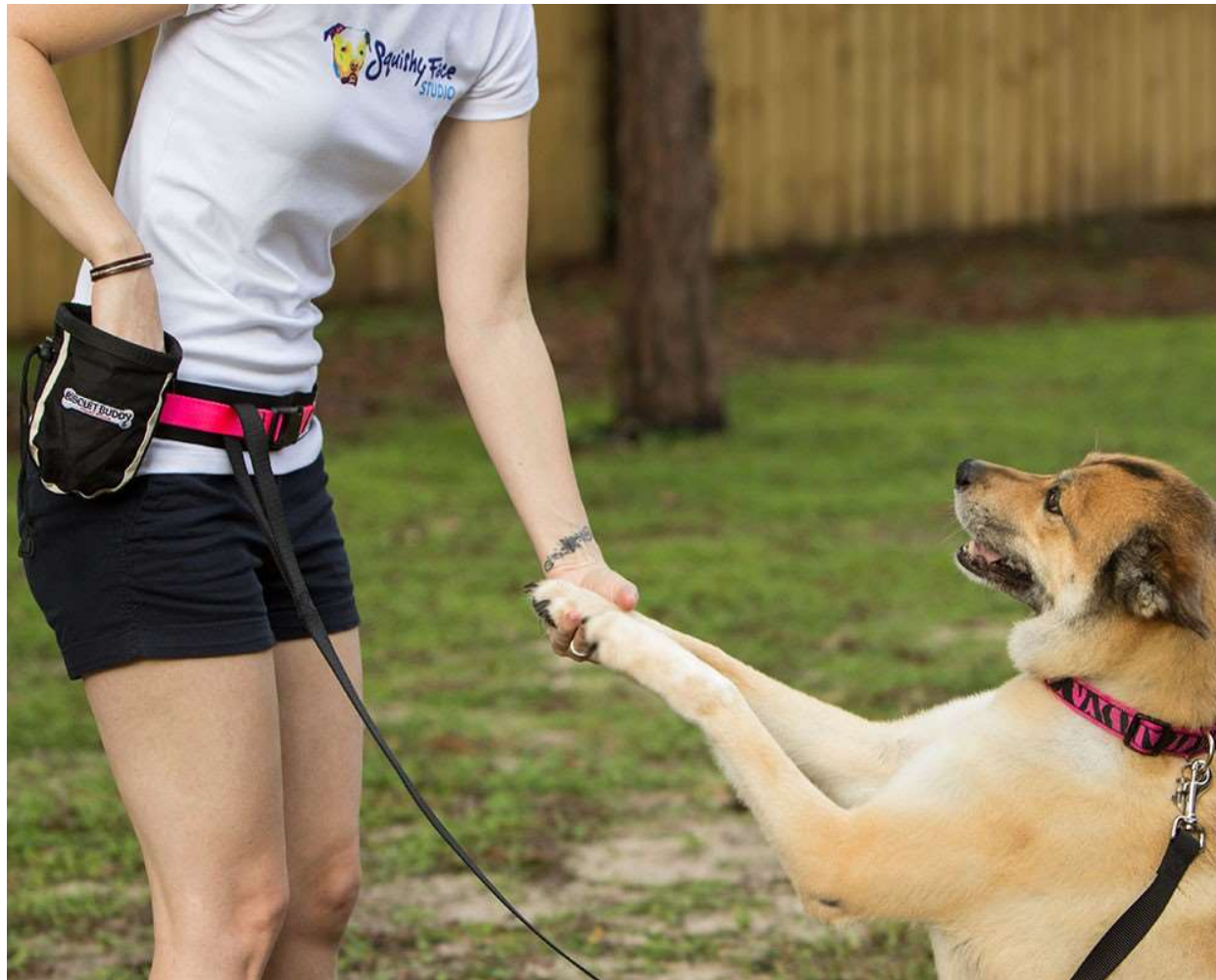
horse

ship

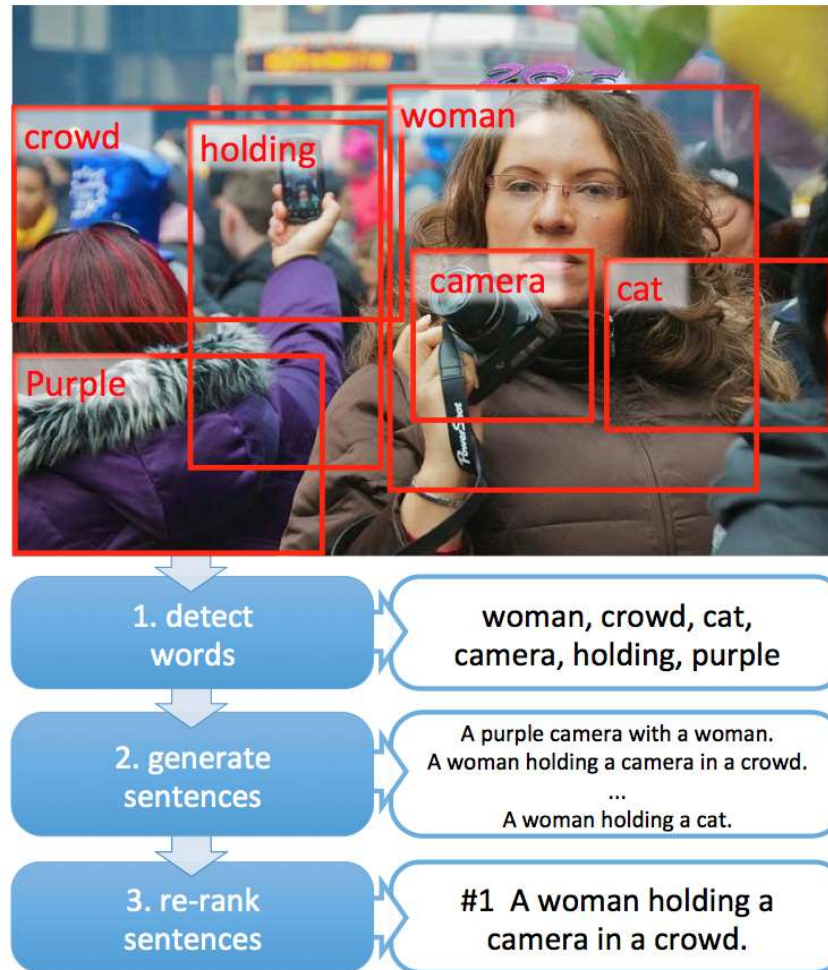
truck



강화 학습 (reinforcement learning)



인공지능의 활용 (이미지 분석)



인공지능의 활용 (이미지 분석)



"man in black shirt is playing guitar."



"construction worker in orange safety vest is working on road."



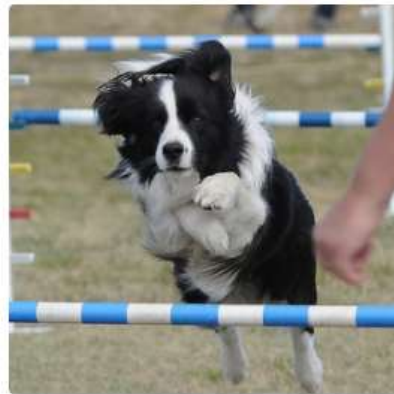
"two young girls are playing with lego toy."



"boy is doing backflip on wakeboard."



"girl in pink dress is jumping in air."



"black and white dog jumps over bar."

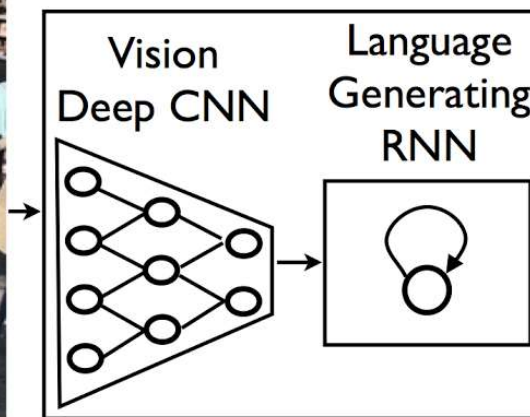


"young girl in pink shirt is swinging on swing."



"man in blue wetsuit is surfing on wave."

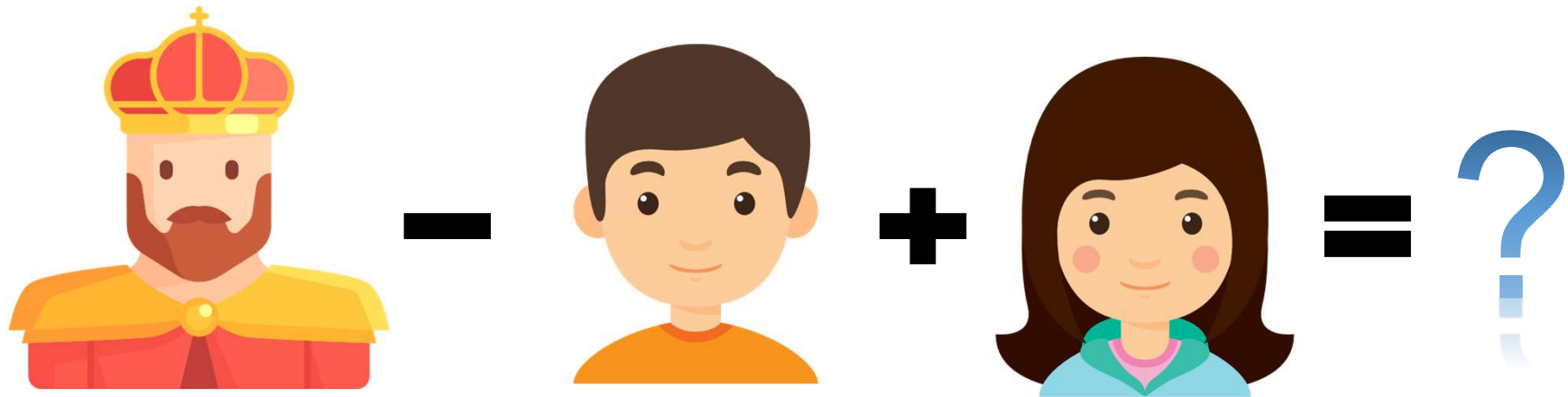
인공지능의 활용 (이미지 분석)



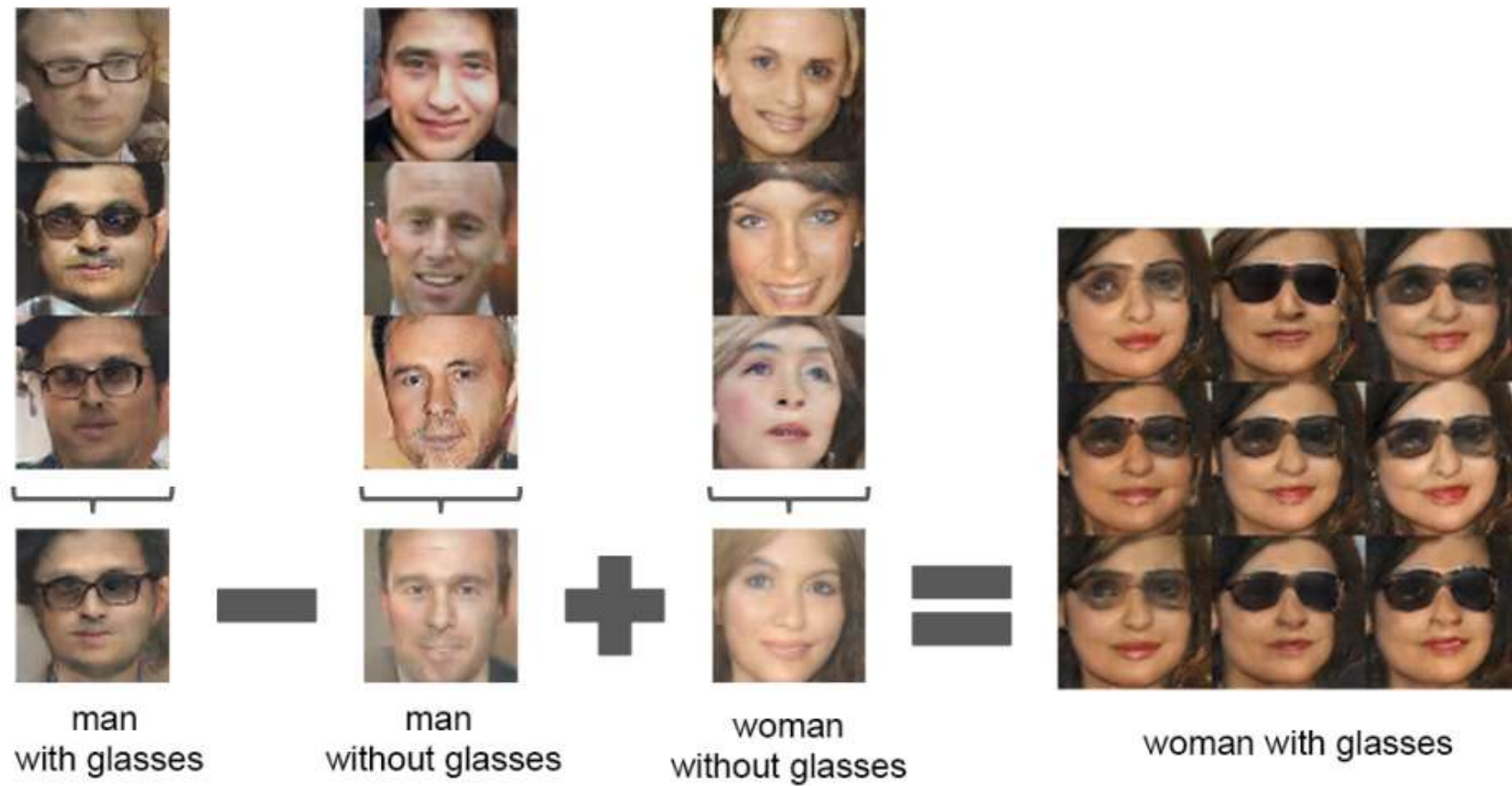
A group of people shopping at an outdoor market.

There are many vegetables at the fruit stand.

인공지능의 활용 (GAN)

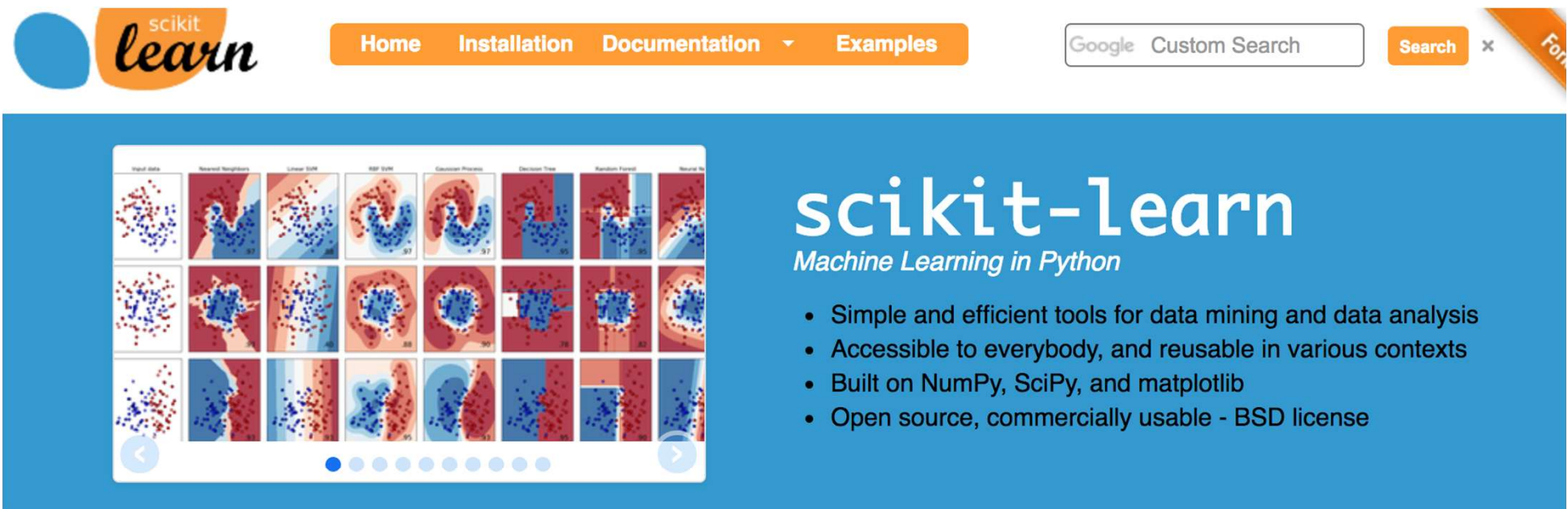


인공지능의 활용 (GAN)



Python을 활용한 기계학습

- <https://github.com/amsukdu/SimplePythonCNN>
- <http://scikit-learn.org/stable/>



Classification

Identifying to which category an object belongs to.

Regression

Predicting a continuous-valued attribute associated with an object.

Clustering

Automatic grouping of similar objects into sets.

결측치의 처리

- 누락된 데이터의 표기 방법
 - null, NaN, NA, ...
- (방법 1) 특징적인 값 사용
 - Ex) -9999, NumPy의 NaN (Not a Number), None
- (방법 2) 별도의 Boolean 배열 사용
 - Ex) empty = [True, False, False, True, ...]

None: Python의 누락된 데이터 표현

- Python에서 누락된 자료를 나타낼 수 있는 가장 일반적인 표현 방법

```
In [1]: import numpy as np  
val = np.array([1, None, 3, 4])  
print(val)  
  
[1 None 3 4]
```

- 단점
 - Python 언어에서 정의된 표현이므로 벡터화된 연산 불가능 → 느린 처리 속도
 - 집계 연산 사용시 오류 발생

```
In [2]: val.sum()
```

TypeError

NaN: 누락된 수치형 데이터 표현

- IEEE 부동 소수점 표기를 사용하는 특수 부동 소수점 값
- 수치형 표현이므로 벡터화된 연산 가능

```
In [1]: import numpy as np
val = np.array([1, np.nan, 3, 4])
print(val)
print(val.sum(), val.min(), val.max())

[ 1. nan  3.  4.]
nan nan nan
```

- 다만 NaN이 포함된 산술 연산의 결과는 또 다른 NaN

```
In [1]: import numpy as np
print(1 + np.nan)
print(0 * np.nan)

nan
nan
```

NumPy에서의 NaN 처리

- 누락된 값을 무시하는 별도의 집계 연산 제공

```
In [1]: import numpy as np
val = np.array([1, np.nan, 3, 4])
print(np.nansum(val), np.nanmin(val), np.nanmax(val))

8.0 1.0 4.0
```

- 주의사항
 - 정수나 문자열 등에서는 NaN에 해당하는 값이 없음

Pandas에서의 NaN과 None

- Pandas에서는 NaN과 None이 거의 호환되며, 적절하게 서로 변환할 수 있게 설계 됨

```
In [1]: import numpy as np
import pandas as pd
val = pd.Series([1, np.nan, 2, None])
print(val)
```

0	1.0
1	NaN
2	2.0
3	NaN

dtype: float64

- None → NaN
- 정수 → 실수

Pandas에서의 NaN과 None

- NaN과 None의 자동 변환 사례
 - None → NaN으로 자동 변환
 - NaN이 포함되면서 int 자료형이 float 자료형으로 변환

```
In [1]: import numpy as np
import pandas as pd
x = pd.Series(range(2), dtype=int)
print(x)
x[0] = None
print(x)
```

```
0    0
1    1
dtype: int64
0    NaN
1    1.0
dtype: float64
```

Pandas에서의 결측치 처리

- 비어있는 값을 감지하고 삭제하고 대체하는 함수 제공
- isnull()
 - 누락된 값들을 가리키는 Boolean 마스크 생성
- notnull()
 - isnull() 함수의 역
- dropna()
 - 데이터에서 비어있는 값을 제거하여 반환
- fillna()
 - 누락된 값을 채운 배열의 사본을 반환

Pandas에서의 결측치 탐지

- isnull()과 notnull() 함수의 사용
- Boolean 배열을 인덱스로 직접 사용 가능

```
In [1]: import numpy as np
import pandas as pd
data = pd.Series([1, np.nan, 'hello', None])
print(data.isnull())
print(data[ data.notnull() ])

0    False
1     True
2    False
3     True
dtype: bool
0         1
2     hello
dtype: object
```


Pandas에서의 결측치 제거 (1)

- dropna() 함수 사용

```
In [1]: import numpy as np
import pandas as pd
data = pd.Series([1, np.nan, 'hello', None])
print(data.dropna())

0      1
2  hello
dtype: object
```

- Series에 대해서는 직관적으로 적용 가능
- DataFrame에서는 단일 값만 삭제할 수 없기 때문에 전체 행이나 전체 열을 삭제

Pandas에서의 결측치 제거 (2)

- DataFrame에서의 dropna()

```
In [1]: import numpy as np
import pandas as pd
df = pd.DataFrame([[1, np.nan, 2],
                   [2, 3, 5],
                   [np.nan, 4, 6]])

print(df)
print(df.dropna())
```

	0	1	2
0	1.0	NaN	2
1	2.0	3.0	5
2	NaN	4.0	6

	0	1	2
1	2.0	3.0	5

- 결측치가 있는 모든 행을 삭제

<https://pastebin.com/XLuPTz6v>

Pandas에서의 결측치 제거 (3)

- dropna()와 축의 지정
 - axis=1로 지정할 경우, 결측치가 있는 모든 열을 삭제

```
In [1]: import numpy as np
import pandas as pd
df = pd.DataFrame([[1, np.nan, 2],
                   [2, 3, 5],
                   [np.nan, 4, 6]])

print(df)
print(df.dropna(axis=1))
```

	0	1	2
0	1.0	NaN	2
1	2.0	3.0	5
2	NaN	4.0	6

	2
0	2
1	5
2	6

Pandas에서의 결측치 제거 (4)

- 일반적인 dropna()는 일부 유효한 데이터도 모두 삭제
- how를 지정하여 옵션 변경 가능
 - any: 기본 옵션
 - all: 모두 빈 경우

```
In [1]: import numpy as np
import pandas as pd
df = pd.DataFrame([[1, np.nan, 2],
                  [2, 3, 5],
                  [np.nan, 4, 6]])

df[3] = np.nan
print(df)
print(df.dropna(axis=1, how='all'))
```

	0	1	2	3
0	1.0	NaN	2	NaN
1	2.0	3.0	5	NaN
2	NaN	4.0	6	NaN

	0	1	2
0	1.0	NaN	2
1	2.0	3.0	5
2	NaN	4.0	6

Pandas에서의 결측치 제거 (4)

- dropna()에 threshold를 지정하여 세부적으로 제어 가능
 - 비어있지 않은 값이 최소 몇 개가 있어야 하는지 지정

```
In [1]: import numpy as np
import pandas as pd
df = pd.DataFrame([[1, np.nan, 2],
                  [2, 3, 5],
                  [np.nan, 4, 6]])
df[3] = np.nan
print(df)
print(df.dropna(axis=0, thresh=3))
```

	0	1	2	3
0	1.0	NaN	2	NaN
1	2.0	3.0	5	NaN
2	NaN	4.0	6	NaN

	0	1	2	3
1	2.0	3.0	5	NaN

Pandas에서의 결측치 채우기 (1)

- 비어있는 값을 삭제하지 않고 유효한 값으로 대체해야 하는 경우 fillna() 함수 사용 가능

```
In [1]: import numpy as np
import pandas as pd
data = pd.Series([1, np.nan, 2, None, 3], index=list('abcde'))
print(data)
print(data.fillna(0))
```

```
a    1.0
b    NaN
c    2.0
d    NaN
e    3.0
dtype: float64
a    1.0
b    0.0
c    2.0
d    0.0
e    3.0
dtype: float64
```

<https://pastebin.com/sPb05BNB>

Pandas에서의 결측치 채우기 (2)

- fillna() 함수를 이용하여 이전 값으로 채우기
 - method를 'ffill'로 지정

```
In [1]: import numpy as np
import pandas as pd
data = pd.Series([1, np.nan, 2, None, 3], index=list('abcde'))

print(data.fillna(method='ffill'))
```

```
a    1.0
b    1.0
c    2.0
d    2.0
e    3.0
dtype: float64
```


Pandas에서의 결측치 채우기 (3)

- fillna() 함수에서 다음 값으로 채우기
 - method를 'bfill'로 지정

```
In [1]: import numpy as np
import pandas as pd
data = pd.Series([1, np.nan, 2, None, 3], index=list('abcde'))

print(data.fillna(method='bfill'))
```

```
a    1.0
b    2.0
c    2.0
d    3.0
e    3.0
dtype: float64
```

Pandas에서의 결측치 채우기 (4)

- DataFrame에서의 fillna() 함수 사용 시 축 지정 가능
 - 아래의 경우 바로 앞 열에 있는 값으로 채움
 - 이전 값을 사용할 수 없는 경우 NaN 값은 그대로 유지

```
In [1]: import numpy as np
import pandas as pd
df = pd.DataFrame([[1, np.nan, 2],
                   [2, 3, 5],
                   [np.nan, 4, 6]])
df[3] = np.nan
print(df.fillna(method='ffill', axis=1))
```

	0	1	2	3
0	1.0	1.0	2.0	2.0
1	2.0	3.0	5.0	5.0
2	NaN	4.0	6.0	6.0

Pandas에서의 결측치 채우기 (5)

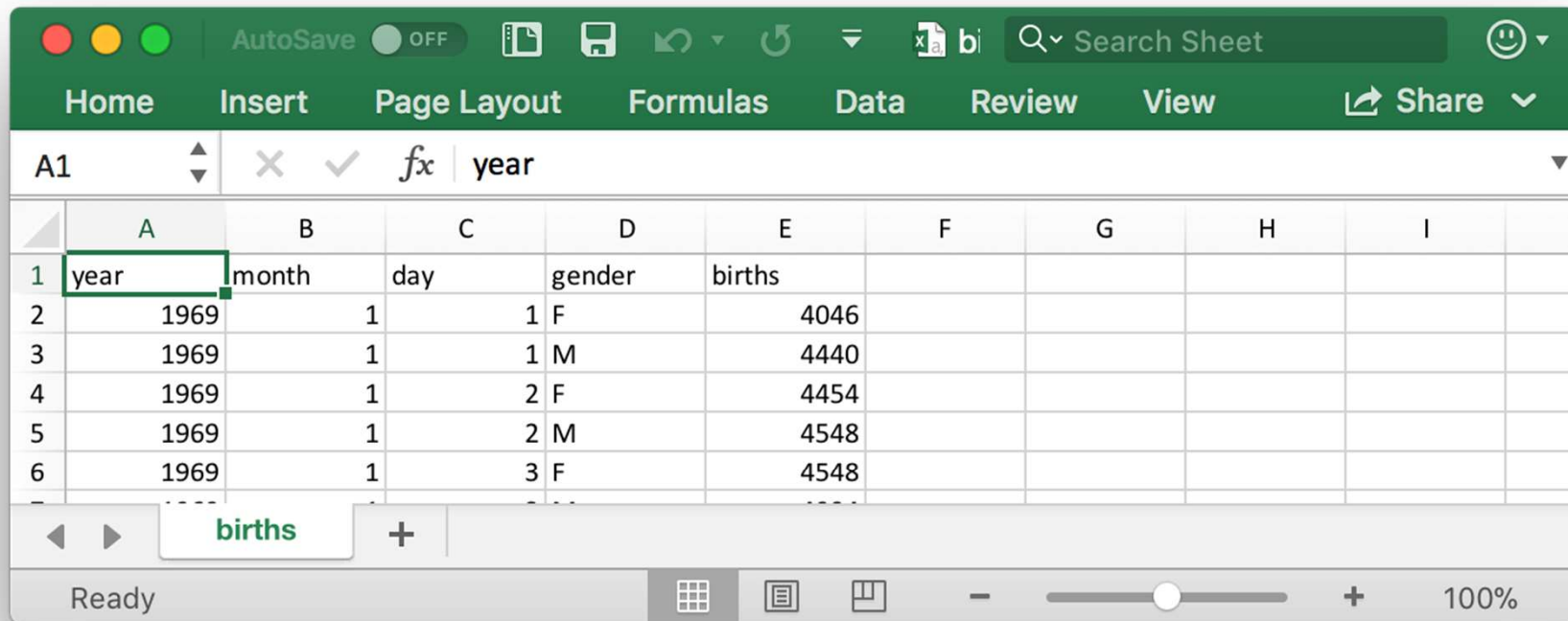
- interpolate() 함수 이용
 - Fill NaN values using an interpolation method

```
>>> s = pd.Series([0, 1, np.nan, 3])
>>> s
0    0.0
1    1.0
2    NaN
3    3.0
dtype: float64
>>> s.interpolate()
0    0.0
1    1.0
2    2.0
3    3.0
dtype: float64
```

```
>>> s = pd.Series([0, 2, np.nan, 8])
>>> s.interpolate(method='polynomial', order=2)
0    0.000000
1    2.000000
2    4.666667
3    8.000000
dtype: float64
```

Pandas를 이용한 파일 입력

- 미국의 출생률 데이터 (births.csv)
 - <https://goo.gl/CQ8jMQ>
 - 작성 중인 IPython 노트북과 같은 폴더에 넣어주세요.



The screenshot shows a Google Sheet interface with a green header bar. The sheet is titled 'births' and has a search bar. The menu bar includes Home, Insert, Page Layout, Formulas, Data, Review, View, and Share. The formula bar shows 'year'. The table has columns A through I. The first row (row 1) is highlighted with a green border and contains the following data: year, month, day, gender, births. The subsequent rows (rows 2 through 6) contain numerical data for the year 1969, month 1, day 1, gender (F, M, F, M, F), and births (4046, 4440, 4454, 4548, 4548).

	A	B	C	D	E	F	G	H	I
1	year	month	day	gender	births				
2	1969	1	1	1 F	4046				
3	1969	1	1	1 M	4440				
4	1969	1	2	2 F	4454				
5	1969	1	2	2 M	4548				
6	1969	1	3	3 F	4548				

Pandas를 이용한 파일 입력

- CSV 파일 입력

```
In [1]: import pandas as pd
        births = pd.read_csv('births.csv')
```

- 데이터의 확인

```
In [2]: import pandas as pd
        births = pd.read_csv('births.csv')
        print(births.head())
```

	year	month	day	gender	births
0	1969	1	1.0	F	4046
1	1969	1	1.0	M	4440
2	1969	1	2.0	F	4454
3	1969	1	2.0	M	4548
4	1969	1	3.0	F	4548

Pandas를 이용한 파일 입력

- 미국의 출생률 데이터 + 파일 소개 (births_a.csv)
 - <https://goo.gl/KKC6oW>
 - 파일의 처음에 부연 설명이 있는 행 존재

```
In [1]: import pandas as pd
        births = pd.read_csv('births_a.csv')
        print(births.head())
```

```
Center for Disease Control birth data Unnamed: 1 Unnamed: 2 Unnamed: 3 \
0      year      month      day      gender
1      1969          1          1          F
2      1969          1          1          M
3      1969          1          2          F
4      1969          1          2          M
```

```
Unnamed: 4
0      births
1      4046
2      4440
3      4454
4      4548
```

Pandas를 이용한 파일 입력

- 건너뛰어야 할 행이 있을 경우 skiprows 지정

```
In [1]: import pandas as pd
births = pd.read_csv('births_a.csv', skiprows=[0])
print(births.head())
```

	year	month	day	gender	births
0	1969	1	1.0	F	4046
1	1969	1	1.0	M	4440
2	1969	1	2.0	F	4454
3	1969	1	2.0	M	4548
4	1969	1	3.0	F	4548

Pandas를 이용한 파일 입력

- Excel 파일의 처리
 - read_excel() 함수 사용
 - sheet_name으로 읽어들이고 sheet의 이름 지정
- 예제 파일 (Niaaa-report.xlsx)
 - <https://goo.gl/gTr17p>

```
In [1]: import pandas as pd
df = pd.read_excel('niaaa-report.xlsx', sheet_name='niaaa-report')
print(df.head())
```

	State	Year	Beer	Wine	Spirits
0	Alabama	1977	0.99	0.13	0.84
1	Alabama	1978	0.98	0.12	0.88
2	Alabama	1979	0.98	0.12	0.84
3	Alabama	1980	0.96	0.16	0.74
4	Alabama	1981	1.00	0.19	0.73

Pandas를 이용한 파일 출력

- DataFrame에 대해서 csv 파일로 바로 저장 가능
 - to_csv() 함수 사용

```
In [1]: import pandas as pd
        births = pd.read_csv('births_a.csv', skiprows=[0])
        births.to_csv('births_b.csv')
```

- 위의 예제에서는 births_a.csv의 설명행을 제외한 나머지 데이터를 births_b.csv에 저장

Pandas를 이용한 파일 출력

- DataFrame에 대해서 excel 파일로도 바로 저장 가능
 - ExcelWriter를 선언하여 저장할 파일 지정
 - to_excel() 함수를 사용

```
In [1]: import pandas as pd
        births = pd.read_csv('births_a.csv', skiprows=[0])
        writer = pd.ExcelWriter('output.xlsx')
        births.to_excel(writer, 'Sheet1')
        writer.save()
```

- ExcelWriter의 save() 함수를 사용해서 내용 저장