3. 데이터 전처리

2022.09.26

Chung-Ang University
AI/ML Innovation Research Center
Hyun-soon Lee

목차

01

데이터 전처리의 기초

01 데이터 전처리의 기초

1. 데이터 전처리의 개념

- 데이터 전처리(data preprocessing) : 머신러닝 모델에 훈련 데이터를 입력하기 전에 데이터를 가공
- **넘파이나 판다스** 같은 **머신러닝의 핵심 도구**,
  맷플롯립과 시본 같은 **데이터 시각화 도구를 활용**하여 실제 데이터를 정리
- 머신러닝 기초 수식

$$y = f(X)$$

  - 데이터 X를 머신러닝 함수 f( )에 넣으면 그 결과 y가 나옴
  - 데이터 X는 훈련 데이터(train data)와 테스트 데이터(test data)가 모두 같은 구조를 갖는 피쳐(feature)이어야 함

2. 데이터 품질 문제

2.1 데이터 분포의 지나친 차이

- 데이터가 연속형 값인데 최댓값과 최솟값 차이가 피쳐보다 더 많이 나는 경우
- 학습에 영향을 줄 수 있기 때문에 데이터의 스케일(scale)을 맞춰줌
  - 데이터의 최댓값과 최솟값을 0에서 1 사이 값으로 바꾸거나 표준 정규분포 형태로 나타내는 등

2.2 기수형 데이터와 서수형 데이터

- 기수형 데이터(하나, 둘, 셋)와 서수형 데이터(첫째, 둘째)는 일반적으로 숫자로 표현되지 않음
- 컴퓨터가 이해할 수 있는 숫자 형태의 정보로 변형

01 데이터 전처리의 기초

2. 데이터 품질 문제

2.3 결측치

- **결측치(missing data)** : 실제로 존재하지만 데이터베이스 등에 기록되지 않는 데이터
- 해당 데이터를 빼고 모델을 돌릴 수 없기 때문에 결측치 처리 전략을 세워 데이터를 채워 넣음
- 판단스(pandas) (4주차 강의)에서 결측치 판단 부분의 처리 부분 참고

2.4 이상치

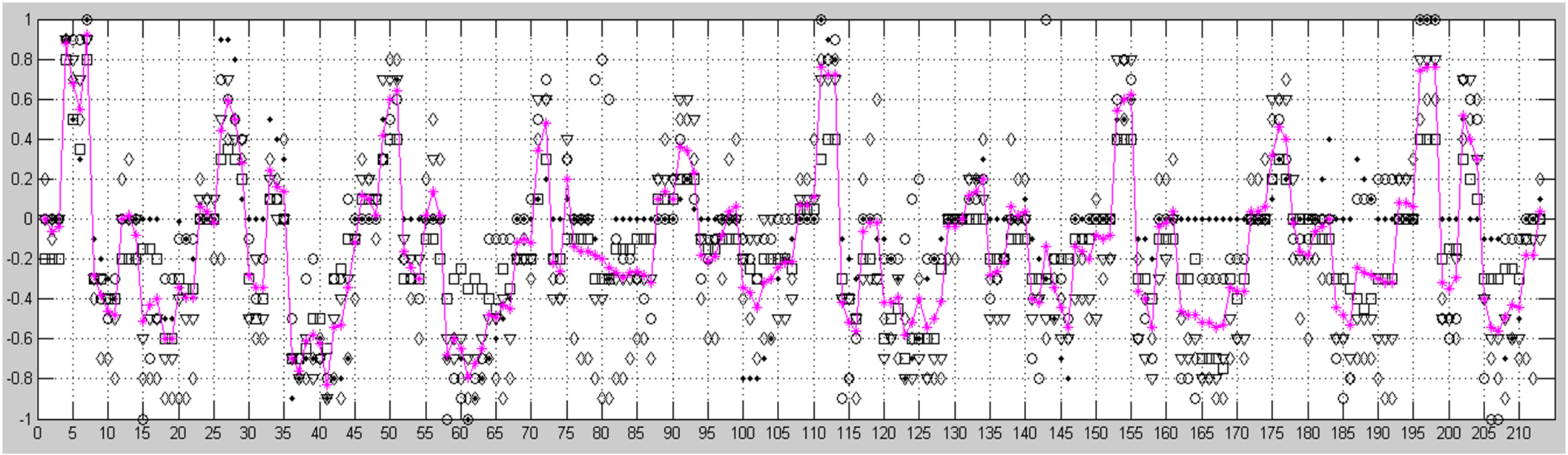
- **이상치(outlier)** : 극단적으로 크거나 작은 값
- 단순히 데이터 분포의 차이와는 다름
- 데이터 오기입이나 특이 현상 때문에 나타남
- 맷플롯립(matplotlib)의 boxplot 함수(4주차 강의)로 판단을 도와 줌

01 데이터 전처리의 기초

2. 데이터 품질 문제

2.4 이상치

- 예제 (원본 데이터)

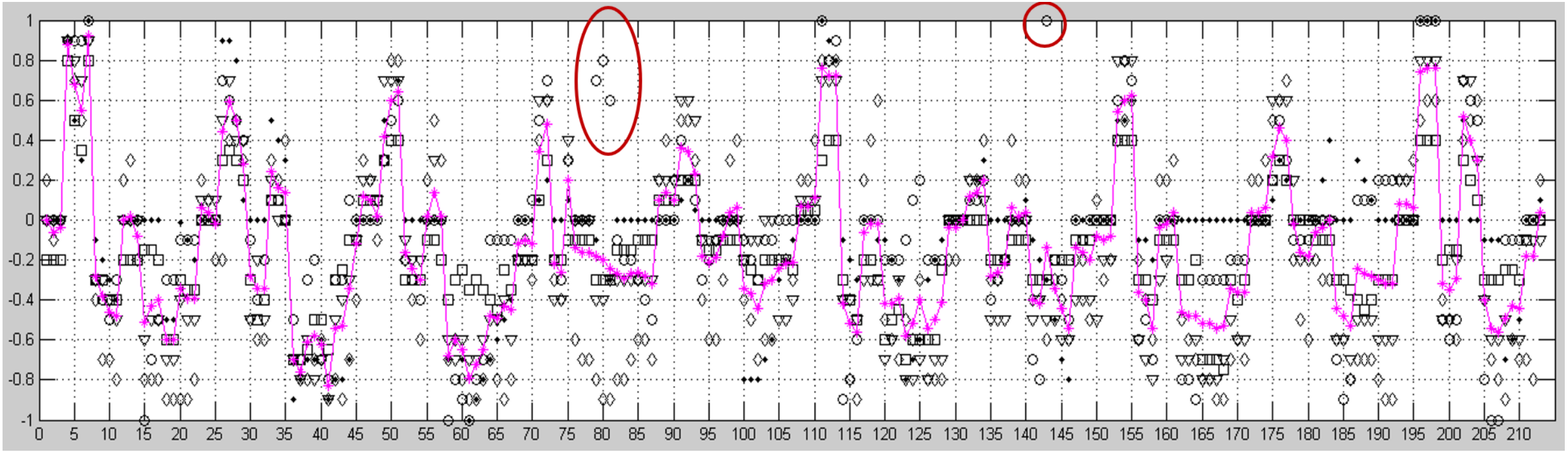


01 데이터 전처리의 기초

2. 데이터 품질 문제

2.4 이상치

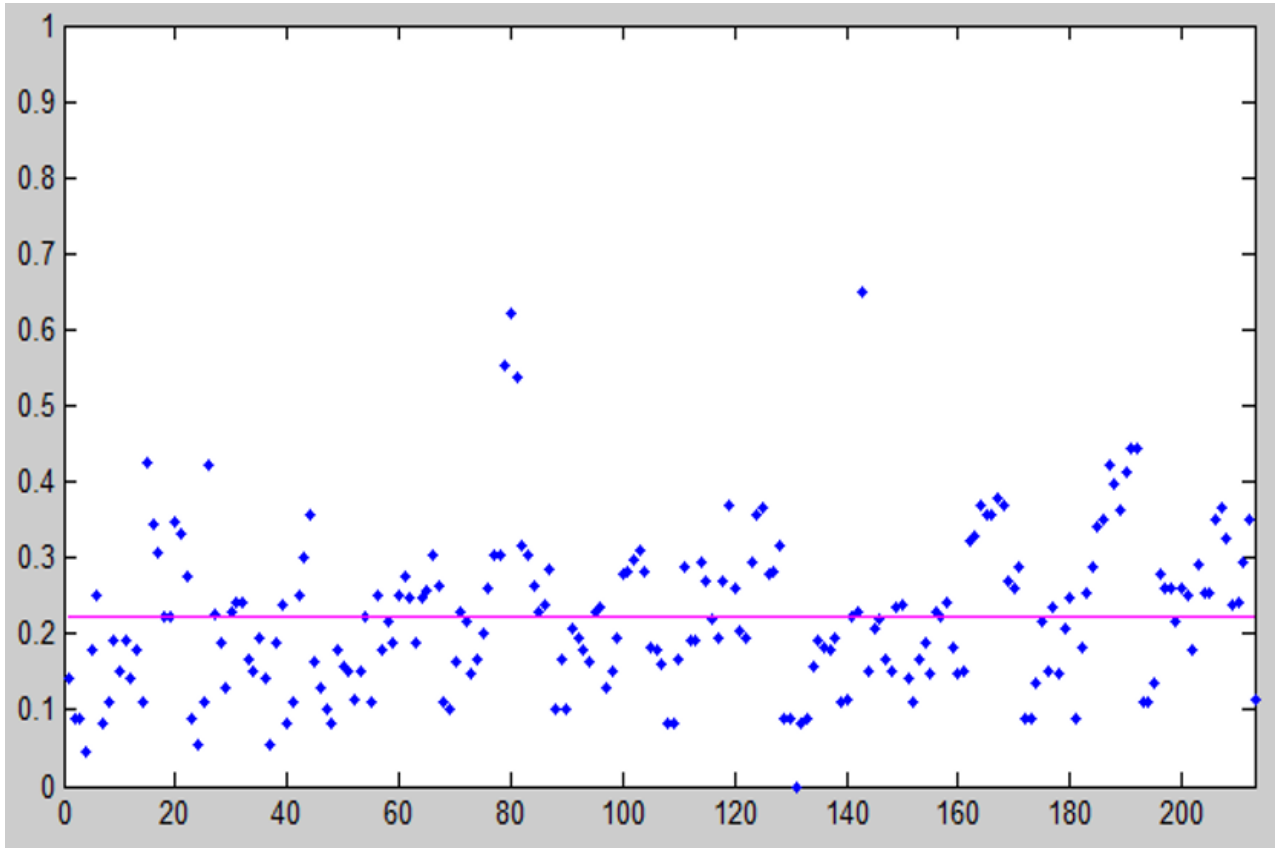
- 예제 (원본 데이터)
- **이상치 데이터 판별은 어떻게 할 것인가?** ○ :표준편차가 큰 데이터

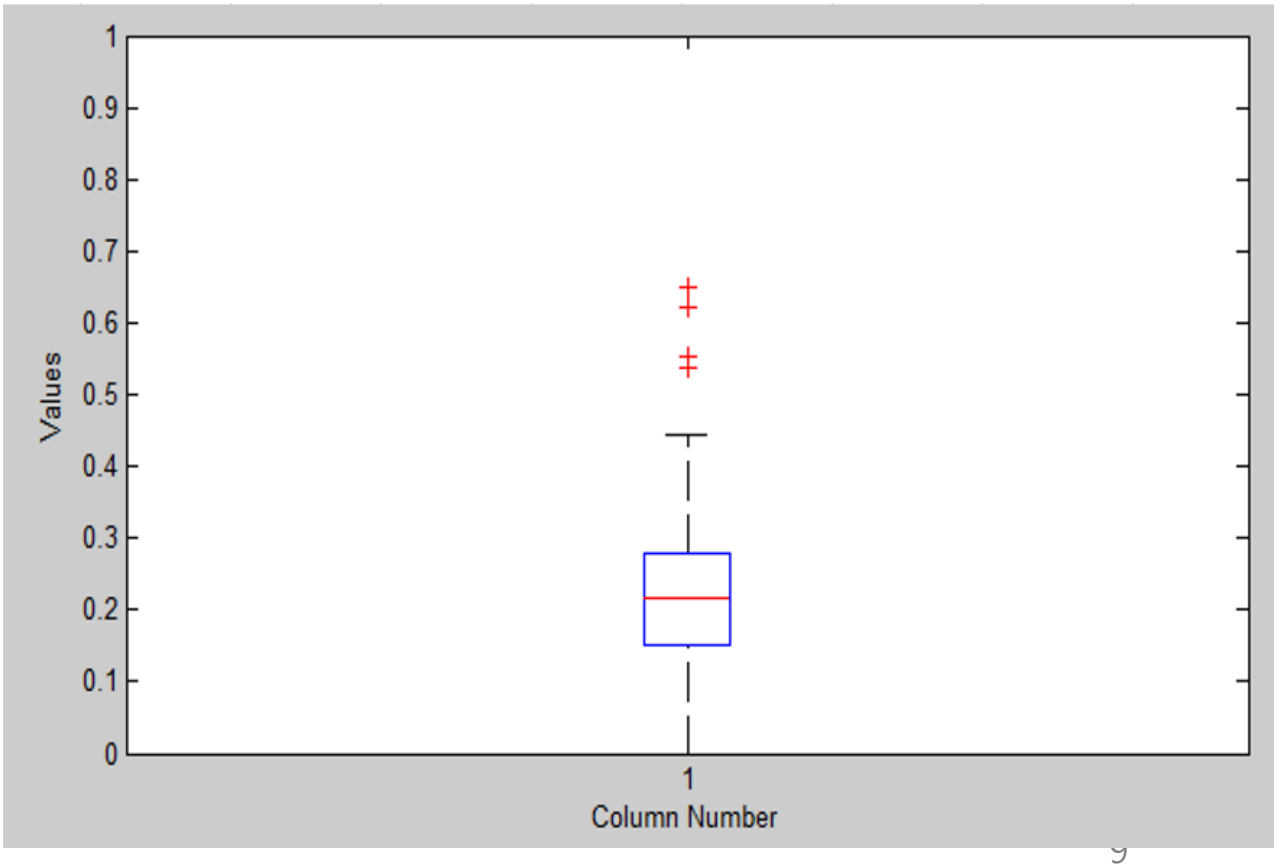


2. 데이터 품질 문제

2.4 이상치

- 예제 (원본 데이터) : 이미지별 평가자 표준편차 (그래프 옆으로 나열 비교 쉬움)





02

데이터 전처리의 전략

02 데이터 전처리의 전략

1. 결측치 처리하기 : 드롭과 채우기

- 데이터를 삭제하거나 데이터를 채움
  - 데이터가 없으면 해당 행이나 열을 삭제
  - 평균값, 최빈값, 중간값 등으로 데이터를 채움

02 데이터 전처리의 전략

1. 결측치 처리하기 : 드롭과 채우기

In [1]:

```
import pandas as pd
import numpy as np

raw_data = {'first_name': ['Jason', np.nan, 'Tina', 'Jake', 'Amy'],
        'last_name': ['Miller', np.nan, 'Ali', 'Milner', 'Cooze'],
        'age': [42, np.nan, 36, 24, 73],
        'sex': ['m', np.nan, 'f', 'm', 'f'],
        'preTestScore': [4, np.nan, np.nan, 2, 3],
        'postTestScore': [25, np.nan, np.nan, 62, 70]}

df = pd.DataFrame(raw_data, columns = ['first_name', 'last_name', 'age', 'sex',
'preTestScore', 'postTestScore'])
df
```

02 데이터 전처리의 전략

1. 결측치 처리하기 : 드롭과 채우기

Out [1]:

	first_name	last_name	age	sex	preTestScore	postTestScore
0	Jason	Miller	42.0	m	4.0	25.0
1	NaN	NaN	NaN	NaN	NaN	NaN
2	Tina	Ali	36.0	f	NaN	NaN
3	Jake	Milner	24.0	m	2.0	62.0
4	Amy	Cooze	73.0	f	3.0	70.0

02 데이터 전처리의 전략

1. 결측치 처리하기 : 드롭과 채우기

- **결측치를 확인할 때 isnull 함수 사용**
  - NaN 값이 존재할 경우 True, 그렇지 않을 경우 False 출력
    - sum 함수로 True인 경우 모두 더하고(sum) 전체 데이터 개수(len)로 나누어 열별 데이터 결측치 비율을 구함

In [2]:	df.isnull().sum() / len(df)
Out [2]:	first_name 0.2 last_name 0.2 age 0.2 sex 0.2 preTestScore 0.4 postTestScore 0.4 dtype: float64

02 데이터 전처리의 전략

1. 결측치 처리하기 : 드롭과 채우기

1.1 드롭

- **드롭(drop) : 결측치가 나온 열이나 행을 삭제**
- **dropna** 사용하여 NaN이 있는 모든 데이터의 행을 없앰

In [3]:

```
df.dropna()
```

Out [3]:

	first_name	last_name	age	sex	preTestScore	postTestScore
0	Jason	Miller	42.0	m	4.0	25.0
3	Jake	Milner	24.0	m	2.0	62.0
4	Amy	Cooze	73.0	f	3.0	70.0

02 데이터 전처리의 전략

1. 결측치 처리하기 : 드롭과 채우기

1.1 드롭

[하나 더 알기] 드롭의 결과물 저장

- 드롭과 관련된 대부분의 명령어들은 실제 드롭한 결과를 반환하나 객체에 드롭 결과를 저장하지는 않음
- 드롭의 결과물을 저장하려면 다른 변수에 재할당
- 또는 매개변수 **inplace=True** 사용
  - 자체적으로 값이 변하면 이후에 해당 데이터를 불러 쓰거나 다시 코드를 실행할 때 문제가 되기 때문에 새로운 값에 복사하는 것이 좋음

In [4]:	**df_no_missing = df.dropna()** **df_no_missing**

02 데이터 전처리의 전략

1. 결측치 처리하기 : 드롭과 채우기

1.1 드롭

- **매개변수 how로 조건에 따라 결측치를 지움**
  - **how**에는 **매개변수 'all'**과 **'any'** 사용
  - 'all'은 행에 있는 모든 값이 NaN일 때 해당 행을 삭제
  - 'any'는 하나의 NaN만 있어도 삭제
- dropna의 **기본 설정은 'any'**라서 모든 결측치를 지움

In [5]:

```
df_cleaned = df.dropna(how='all')
df_cleaned
```

Out [5]:

	first_name	last_name	age	sex	preTestScore	postTestScore
0	Jason	Miller	42.0	m	4.0	25.0
2	Tina	Ali	36.0	f	NaN	NaN
3	Jake	Milner	24.0	m	2.0	62.0
4	Amy	Cooze	73.0	f	3.0	70.0

02 데이터 전처리의 전략

1. 결측치 처리하기 : 드롭과 채우기

1.1 드롭

- 열 값이 모두 NaN일 경우에는 축(axis)을 추가하여 삭제
  - location이라는 열을 추가하여 값들을 모두 NaN으로 한 후 axis=1(axis=1: 열 기준, axis=0:행 기준)로 location 열만 삭제

In [6]:

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

Out [6]:

	first_name	last_name	age	sex	preTestScore	postTestScore
0	Jason	Miller	42.0	m	4.0	25.0
1	NaN	NaN	NaN	NaN	NaN	NaN
2	Tina	Ali	36.0	f	NaN	NaN
3	Jake	Milner	24.0	m	2.0	62.0
4	Amy	Cooze	73.0	f	3.0	70.0

02 데이터 전처리의 전략

1. 결측치 처리하기 : 드롭과 채우기

1.1 드롭

- 매개변수 thresh 데이터의 개수를 기준으로 삭제
  - **thresh=1** 지정하면 데이터가 한 개라도 존재하는 행은 남김

In [7]:

```
df.dropna(axis=0, thresh=1)
```

Out [7]:

	first_name	last_name	age	sex	preTestScore	postTestScore	location
0	Jason	Miller	42.0	m	4.0	25.0	NaN
2	Tina	Ali	36.0	f	NaN	NaN	NaN
3	Jake	Milner	24.0	m	2.0	62.0	NaN
4	Amy	Cooze	73.0	f	3.0	70.0	NaN

02 데이터 전처리의 전략

1. 결측치 처리하기 : 드롭과 채우기

1.1 드롭

- thresh=5 지정하면 데이터가 다섯 개 이상 있어야 남김

In [8]:
```
df.dropna(thresh=5)
```

Out [8]:

	first_name	last_name	age	sex	preTestScore	postTestScore	location
0	Jason	Miller	42.0	m	4.0	25.0	NaN
3	Jake	Milner	24.0	m	2.0	62.0	NaN
4	Amy	Cooze	73.0	f	3.0	70.0	NaN

02 데이터 전처리의 전략

1. 결측치 처리하기 : 드롭과 채우기

1.2 채우기

- **채우기(fill)** : 비어있는 값을 채움
- 일반적으로 드롭한 후에 남은 값들을 채우기 처리
- 평균, 최빈값 등 데이터의 분포를 고려해서 채움
- **함수 fillna** 사용

In [9]:

```
df.fillna(0)
```

Out [9]:

	first_name	last_name	age	sex	preTestScore	postTestScore	location
0	Jason	Miller	42.0	m	4.0	25.0	0.0
1	0	0	0.0	0	0.0	0.0	0.0
2	Tina	Ali	36.0	f	0.0	0.0	0.0
3	Jake	Milner	24.0	m	2.0	62.0	0.0
4	Amy	Cooze	73.0	f	3.0	70.0	0.0

02 데이터 전처리의 전략

1. 결측치 처리하기 : 드롭과 채우기

1.2 채우기

- 빈 값에 평균값을 채우려면 열 단위의 평균값을 계산하여 해당 열에만 값을 채움
  - **매개변수 inplace**는 변경된 값을 리턴시키는 것이 아니고 **해당 변수 자체의 값을 변경**

In [10]:

```
df["preTestScore"].fillna(df["preTestScore"].mean(), inplace=True)
df
```

Out [10]:

	first_name	last_name	age	sex	preTestScore	postTestScore	location
0	Jason	Miller	42.0	m	4.0	25.0	NaN
1	NaN	NaN	NaN	NaN	3.0	NaN	NaN
2	Tina	Ali	36.0	f	3.0	NaN	NaN
3	Jake	Milner	24.0	m	2.0	62.0	NaN
4	Amy	Cooze	73.0	f	3.0	70.0	NaN

02 데이터 전처리의 전략

1. 결측치 처리하기 : 드롭과 채우기

1.2 채우기

- 열별 분포를 고려하여 채울 수 있음
  - groupby 함수로 각 인덱스의 성별에 따라 빈칸을 채움

In [11]:	df.groupby("sex")["postTestScore"].transform("mean")
Out [11]:	0 43.5 1 NaN 2 70.0 3 43.5 4 70.0 Name: postTestScore, dtype: float64

02 데이터 전처리의 전략

1. 결측치 처리하기 : 드롭과 채우기

1.2 채우기

- fillna 함수 안에 transform을 사용하여 인덱스를 기반으로 채울 수 있음
  - 일반적으로 쓰이는 기법

In [12]:

```
df["postTestScore"].fillna(
    df.groupby("sex")["postTestScore"].transform("mean"),
inplace=True)
df
```

Out [12]:

	first_name	last_name	age	sex	preTestScore	postTestScore	location
0	Jason	Miller	42.0	m	4.0	25.0	NaN
1	NaN	NaN	NaN	NaN	3.0	NaN	NaN
2	Tina	Ali	36.0	f	3.0	70.0	NaN
3	Jake	Milner	24.0	m	2.0	62.0	NaN
4	Amy	Cooze	73.0	f	3.0	70.0	NaN

02 데이터 전처리의 전략

2. 범주형 데이터 처리하기 : 원핫인코딩

- **원핫인코딩(one-hot encoding)** : 범주형 데이터의 개수만큼 **가변수(dummy variable)를 생성**하여 존재 유무를 1 또는 0으로 표현
  - color라는 변수에 {Green, Blue, Yellow} 3개의 값이 있을 때
  - 3개의 가변수를 만들고 각 색상에 인덱스를 지정
    - Green의 인덱스는 0, Blue의 인덱스 1, Yellow의 인덱스는 2로 지정
  - 해당 값이면 1, 아니면 0을 입력

$$\{Green\} \rightarrow [1, 0, 0]$$

$$\{Blue\} \rightarrow [0, 1, 0]$$

$$\{Yellow\} \rightarrow [0, 0, 1]$$

02 데이터 전처리의 전략

2. 범주형 데이터 처리하기 : 원핫인코딩

- 원핫인코딩을 적용하려면 **판다스**에서 제공하는 **get_dummies 함수**를 이용하거나 **사이킷런(scikit-learn)**에서 제공하는 **LabelEncoder**나 **OneHotEncoder**를 이용

In [13]:

```
edges = pd.DataFrame({'source': [0, 1, 2],
'target': [2, 2, 3],
'weight': [3, 4, 5],
'color': ['red', 'blue', 'blue']})
edges
```

Out [13]:

	source	target	weight	color
0	0	2	3	red
1	1	2	4	blue
2	2	3	5	blue

02 데이터 전처리의 전략

2. 범주형 데이터 처리하기 : 원핫인코딩

- **get_dummies**를 적용하여 범주형 데이터 color에 가변수 추가

In [14]:

```
edges.dtypes
```

Out [14]:

```
source int64
target int64
weight int64
color object
dtype: object
```

In [15]:

```
pd.get_dummies(edges)
```

Out [15]:

	source	target	weight	color_blue	color_red
0	0	2	3	0	1
1	1	2	4	1	0
2	2	3	5	1	0

02 데이터 전처리의 전략

2. 범주형 데이터 처리하기 : 원핫인코딩

In [16]:
```
pd.get_dummies(edges["color"])
```

Out [16]:

	blue	red
0	0	1
1	1	0
2	1	0

In [17]:
```
pd.get_dummies(edges[["color"]])
```

Out [17]:

	color_blue	color_red
0	0	1
1	1	0
2	1	0

02 데이터 전처리의 전략

2. 범주형 데이터 처리하기 : 원핫인코딩

- 필요에 따라 정수형을 객체로 변경해서 처리

	source	target	weight	color
0	0	2	3	red
1	1	2	4	blue
2	2	3	5	blue

그림 6-1 다음 코드에서 다룰 데이터

- weight는 숫자로 되어 있지만 기수형 데이터
- 데이터를 M, L, XL로 변경하여 원핫인코딩을 적용

02 데이터 전처리의 전략

2. 범주형 데이터 처리하기 : 원핫인코딩

- 데이터를 원핫인코딩 형태로 변경한 후 필요에 따라 병합이나 연결로 두 가지의 데이터를 합침

In [18]:

```
weight_dict = {3:"M", 4:"L", 5:"XL"}
edges["weight_sign"] = edges["weight"].map(weight_dict)
weight_sign = pd.get_dummies(edges["weight_sign"])
weight_sign
```

Out [18]:

	L	M	XL
0	0	1	0
1	1	0	0
2	0	0	1

Out [19]:

	source	target	weight	color	weight_sign	L	M	XL
0	0	2	3	red	M	0	1	0
1	1	2	4	blue	L	1	0	0
2	2	3	5	blue	XL	0	0	1

In [19]:

```
pd.concat([edges, weight_sign], axis=1)
```

02 데이터 전처리의 전략

3. 범주형 데이터로 변환하여 처리하기 : 바인딩

- **바인딩(binding)** : 연속형 데이터를 범주형 데이터로 변환

In [20]:

```
raw_data = {'regiment': ['Nighthawks', 'Nighthawks', 'Nighthawks', 'Nighthawks',
'Dragoons', 'Dragoons', 'Dragoons', 'Dragoons', 'Scouts', 'Scouts', 'Scouts', 'Scouts'],
        'company': ['1st', '1st', '2nd', '2nd', '1st', '1st', '2nd', '2nd','1st', '1st', '2nd', '2nd'],
        'name': ['Miller', 'Jacobson', 'Ali', 'Milner', 'Cooze', 'Jacon', 'Ryaner', 'Sone',
'Sloan', 'Piger', 'Riani', 'Ali'],
        'preTestScore': [4, 24, 31, 2, 3, 4, 24, 31, 2, 3, 2, 3],
        'postTestScore': [25, 94, 57, 62, 70, 25, 94, 57, 62, 70, 62, 70]}

df = pd.DataFrame(raw_data, columns = ['regiment', 'company', 'name',
'preTestScore', 'postTestScore'])
df
```

02 데이터 전처리의 전략

3. 범주형 데이터로 변환하여 처리하기 : 바인딩

Out [20]:

	regiment	company	name	preTestScore	postTestScore
0	Nighthawks	1st	Miller	4	25
1	Nighthawks	1st	Jacobson	24	94
2	Nighthawks	2nd	Ali	31	57
3	Nighthawks	2nd	Milner	2	62
4	Dragoons	1st	Cooze	3	70
5	Dragoons	1st	Jacon	4	25
6	Dragoons	2nd	Ryaner	24	94
7	Dragoons	2nd	Sone	31	57
8	Scouts	1st	Sloan	2	62
9	Scouts	1st	Piger	3	70
10	Scouts	2nd	Riani	2	62
11	Scouts	2nd	Ali	3	70

02 데이터 전처리의 전략

3. 범주형 데이터로 변환하여 처리하기 : 바인딩

- postTestScore에 대한 학점을 측정하는 코드를 작성
  - 데이터 범위를 구분 : 0~25, 25~50, 50~75, 75~100으로 구분
  - 함수 cut 사용
    - bins 리스트에 구간의 시작 값 끝 값을 넣고 구간의 이름을 리스트로 나열 bins의 원소는 5개이고 group_names는 4개
    - cut 함수로 나눌 시리즈 객체와 구간, 구간의 이름을 넣어주면 해당 값을 바인딩하여 표시해줌

02 데이터 전처리의 전략

3. 범주형 데이터로 변환하여 처리하기 : 바인딩

In [21]:
```
bins = [0, 25, 50, 75, 100]      # bins 정의(0-25, 25-50, 60-75, 75-100)
group_names = ['Low', 'Okay', 'Good', 'Great']
categories = pd.cut(
    df['postTestScore'], bins, labels=group_names)
categories
```

02 데이터 전처리의 전략

3. 범주형 데이터로 변환하여 처리하기 : 바인딩

Out [21]:	0 Low 1 Great 2 Good 3 Good 4 Good 5 Low 6 Great 7 Good 8 Good 9 Good 10 Good 11 Good Name: postTestScore, dtype: category Categories (4, object): ['Low' < 'Okay' < 'Good' < 'Great']

4. 데이터의 크기 맞추기 : 피쳐 스케일링

- **스케일링(scaling)** : 데이터 간 범위를 맞춤
  - 몸무게와 키를 하나의 모델에 넣으면 데이터의 범위가 훨씬 넓어져 키가 몸무게에 비해 모델에 과다하게 영향을 줌
- x1과 x2의 변수 범위가 다를 때 하나의 변수 범위로 통일시켜 처리

그림 6-2 x_1과 x_2의 변수 범위의 차이

4. 데이터의 크기 맞추기 : 피쳐 스케일링

- **최솟값-최댓값 정규화(min-max normalization)** :
  최솟값과 최댓값을 기준으로 0에서 1, 또는 0에서 지정 값까지로 값의 크기를 변화시킴

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}(new_{\max} - new_{\min}) - new_{\max}$$

  - x는 처리하고자 하는 열, x_i는 이 열의 하나의 값,
    max(x)는 해당 열의 최댓값, min(x)는 해당 열의 최솟값
  - new_max와 new_min은 새롭게 지정되는 값의 최댓값 또는 최솟값

4. 데이터의 크기 맞추기 : 피쳐 스케일링

- **z-스코어 정규화(z-score normalization)** :
  기존 값을 표준 정규분포값으로 변환하여 처리

$$z = \frac{x_i - \mu}{\sigma}$$

  - μ는 x 열의 평균값이고 σ는 표준편차
  - 통계학 시간에 배우는 수식과 동일

02 데이터 전처리의 전략

4. 데이터의 크기 맞추기 : 피쳐 스케일링

In [22]:

```
df = pd.DataFrame(
    {'A':[14.00,90.20,90.95,96.27,91.21],
     'B':[103.02,107.26,110.35,114.23,114.68],
     'C':['big','small','big','small','small']})

df
```

Out [22]:

	A	B	C
0	14.00	103.02	big
1	90.20	107.26	small
2	90.95	110.35	big
3	96.27	114.23	small
4	91.21	114.68	small

4. 데이터의 크기 맞추기 : 피쳐 스케일링

- 스케일링할 때는 브로드캐스팅 개념으로 스칼라 값(평균값, 최댓값, 최솟값)과 벡터(열) 값 간 연산

In [23]:	df["A"] - df["A"].min() #A열에서 A열의 최솟값을 뺀다.
Out [23]:	0 0.00 1 76.20 2 76.95 3 82.27 4 77.21 Name: A, dtype: float64

4. 데이터의 크기 맞추기 : 피쳐 스케일링

- 최솟값-최댓값 정규화 방법에서 최댓값과 최솟값을 따로 구하지 않고 코드로 수식을 나타낼 수 있음

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

In [24]:	(df["A"] - df["A"].min()) / (df["A"].max() -df["A"].min())
Out [24]:	0 0.000000 1 0.926219 2 0.935335 3 1.000000 4 0.938495 Name: A, dtype: float64

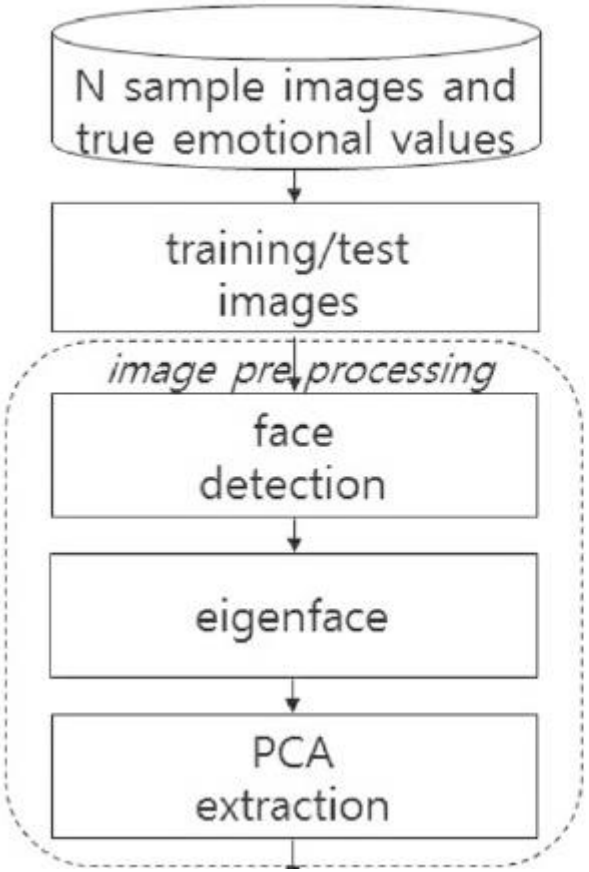
4. 데이터의 크기 맞추기 : 피쳐 스케일링

- **z-스코어 정규화 수식** 역시 코드로 나타낼 수 있음

In [25]:	(df["B"] - df["B"].mean()) / (df["B"].std())
Out [25]:	0 -1.405250 1 -0.540230 2 0.090174 3 0.881749 4 0.973556 Name: B, dtype: float64

02 데이터 전처리의 전략

*적용 예제1[1]



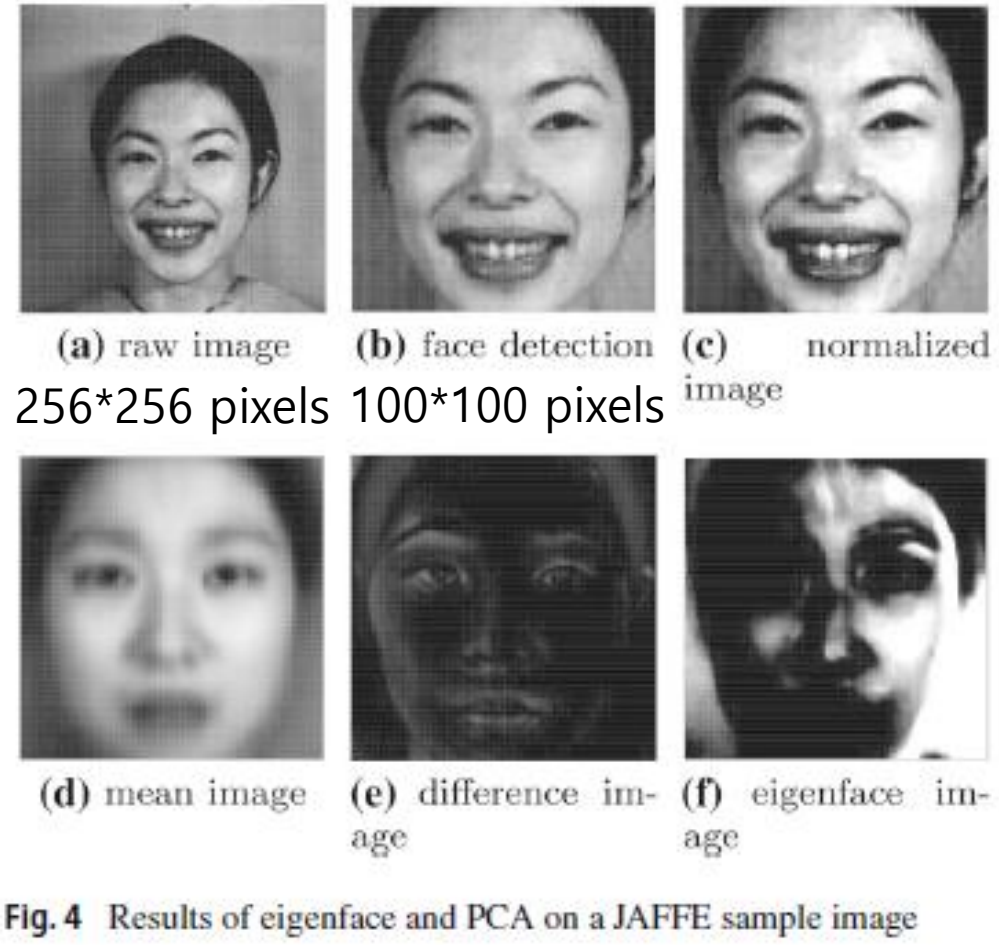


Fig. 4 Results of eigenface and PCA on a JAFFE sample image

[1] LEE, Hyun-Soon; KANG, Bo-Yeong. Continuous emotion estimation of facial expressions on JAFFE and CK+ datasets for human–robot interaction. *Intelligent service robotics*, 2020, 13.1: 15-27.

02 데이터 전처리의 전략

*적용 예제2[2]

- 적외선 센서(GP2Y0A21YK)들의 잡음 제거를 위한 전저리 과정

Butterworth filter:

$$N(\omega) = \frac{1}{\sqrt{1+\omega^{2n}}}$$

Figure 4. Positions of infrared sensors on the center console.

[2] LEE, Hyun-Soon, et al. Estimation of driver's danger level when accessing the center console for safe driving. *Sensors*, 2018, 18.10: 3392.

03

데이터 전처리의 실습

03 데이터 전처리의 실습

1. 머신러닝 프로세스와 데이터 전처리

- 데이터를 확보한 후 데이터를 정제 및 전처리
- 학습용과 테스트 데이터를 나눠 학습용 데이터로 학습을 실시
- 학습 결과를 평가 지표와 비교하여 하이퍼 매개변수 변환
- 최종적인 모델 생성하여 테스트 데이터셋으로 성능을 측정
- 모델을 시스템에 배치하여 모델을 작동시킴

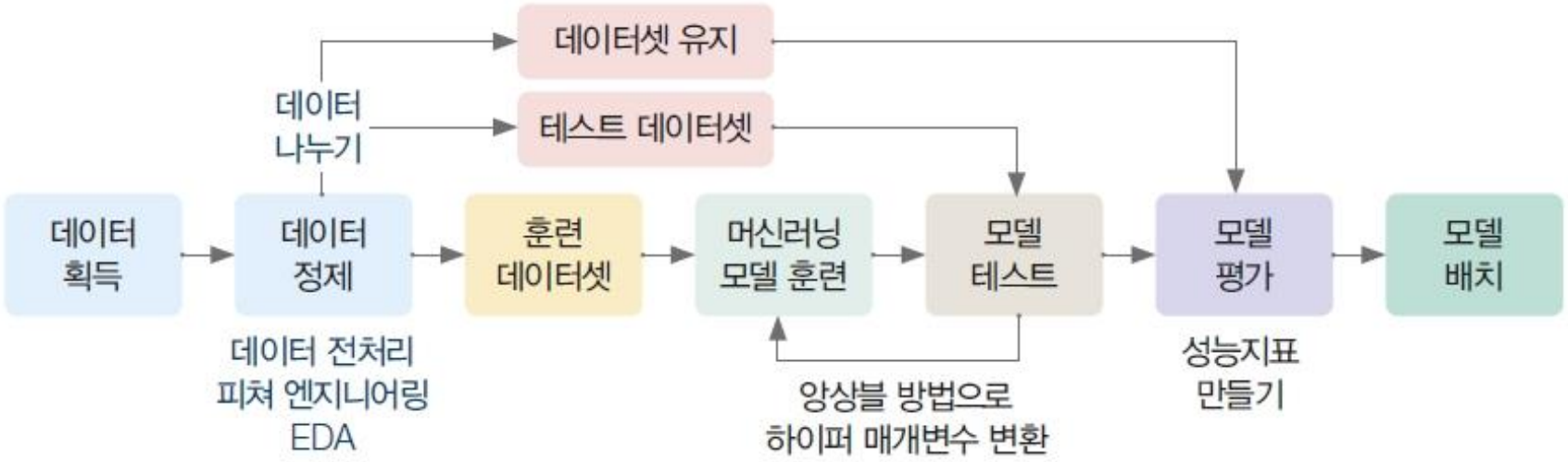


그림 6-3 머신러닝 프로세스

03 데이터 전처리의 실습

1. 머신러닝 프로세스와 데이터 전처리

- 데이터 정제/데이터 전처리 단계는 실제로 가장 많은 시간이 들어가는 작업
  - 머신러닝 모델을 만드는 'ML Code' 작업은 가장 작은 부분을 차지하고, 데이터 수집이나 피쳐 추출(Feature Extraction), 데이터 검증(Data Verification)이 훨씬 더 많은 부분을 차지

그림 6-4 머신러닝 시스템 개발의 업무 비중 © D Scully

03 데이터 전처리의 실습

1. 머신러닝 프로세스와 데이터 전처리

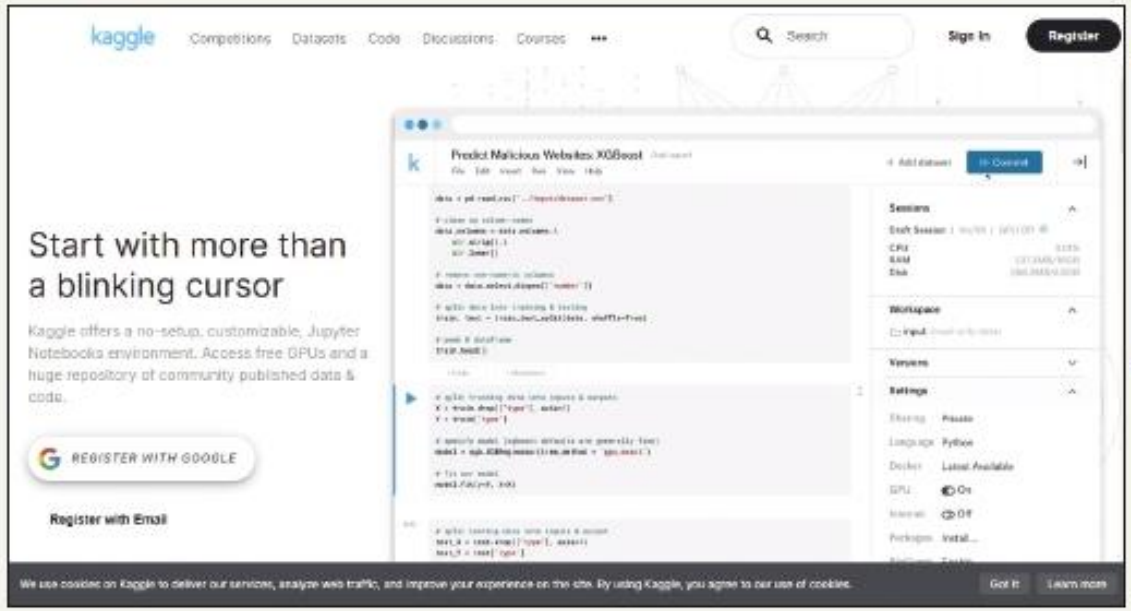
[하나 더 알기] 데이터 확보를 위한 최적의 장소 : 캐글과 데이콘

- 실제 회사에서 생산 및 처리하고 있는 데이터를 확보한다면 최적이겠지만, 이러한 실제 데이터를 확보하기는 어렵다.
- 다행히 많은 엔지니어들이 연습용 데이터를 제공하고 있다. 대표적으로 캐글(Kaggle)과 데이콘(DACON)이 있다.
- **캐글**은 **2017년에 구글에 인수되면서 사실상 데이터 분석의 표준적인 프레임워크**로 사용되고 있고, **데이콘**은 **국내 스타트업이 운영하는 데이터 대회 사이트**이다.
- 두 사이트 모두 기본적으로 같은 구조를 가지고 있으며 교육과 코드 공유를 통한 데이터 분석 커뮤니티 발전에 도움을 주고 있다.
- 처음 데이터 분석을 배우면서 1차적으로 어느 정도 정리된 캐글과 데이콘의 데이터를 활용하는 것이 좋다.

03 데이터 전처리의 실습

1. 머신러닝 프로세스와 데이터 전처리

[하나 더 알기] 데이터 확보를 위한 최적의 장소 : 캐글과 데이콘



(a) 캐글(Kaggle)의 웹사이트 : https://www.kaggle.com/

(b) 데이콘(DACON)의 웹사이트 : https://dacon.io/

그림 6-5 캐글(Kaggle)과 데이콘(DACON)

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

- 타이타닉 문제는 캐글(Kaggle)에 있는 많은 데이터 중 데이터 분석 입문자가 처음 사용하기 좋은 데이터
- 데이터가 기본적이면서 평가가 쉬움

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

[하나 더 알기] 타이타닉 웹페이지

- 개요, 데이터, 코드, 논의, 리더보드, 규칙 등으로 구성
- 처음에는 개요 페이지에서 대회의 내용과 평가지표를 확인
- 평가지표에 맞게 머신러닝 모델링을 실시
- **타이타닉 문제는 배에 타고 있는 승객 대비 살아남을 수 있는 승객을 예측하는 모델**로, 'accuracy'라는 지표를 사용

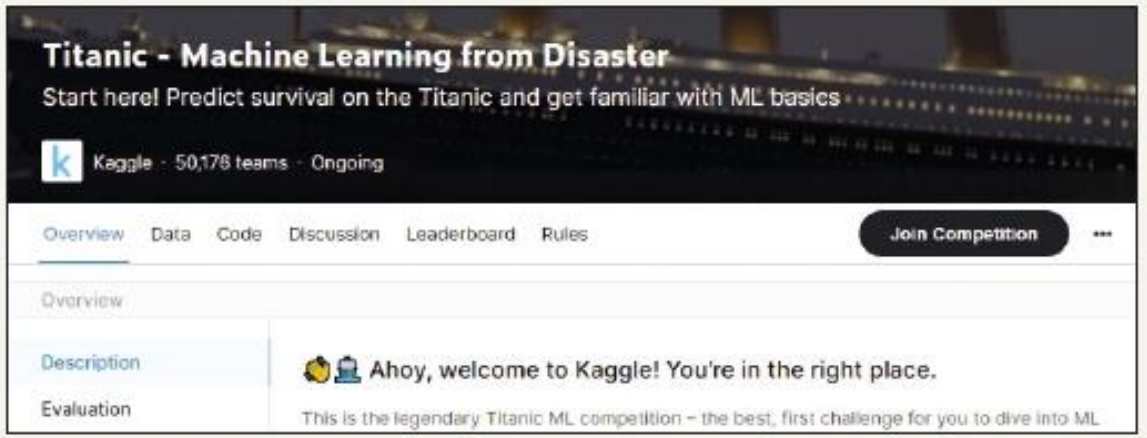


그림 6-6 캐글(Kaggle)의 타이타닉 웹페이지

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.1 데이터 확보하기

- https://www.kaggle.com/c/titanic
- [Data] 탭 - 왼쪽 하단 [Download All] 버튼

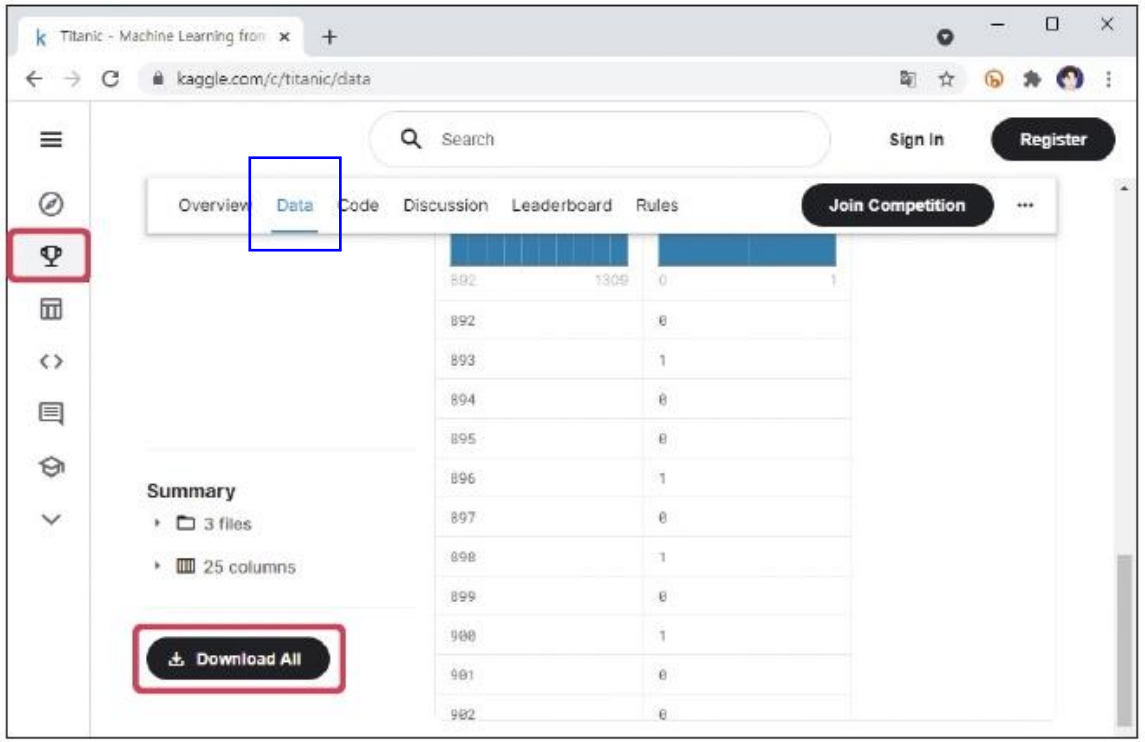


그림 6-7 타이타닉 데이터 다운로드하기

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.1 데이터 확보하기

- ‘titanic.zip’ 파일의 압축을 풂
  - gender_submission.csv : 데이터 제출 예제 파일로 캐글에 제출하여 평가를 받을 파일의 예시
  - test.csv : 예측되는 탑승객들의 데이터가 있는 파일
  - train.csv : 모델을 학습시키기 위한 데이터가 있는 파일
  - ‘train.csv’ 파일 데이터를 사용하여 모델을 만들고 모델을 ‘test.csv’ 데이터에 적용하여 결과를 ‘gender_submission.csv’ 파일 형태로 제출

그림 6-8 titanic.zip 파일에 있는 3개의 데이터 파일

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.1 데이터 확보하기

- 'train.csv' 파일과 달리 'test.csv' 파일에는 y 값, 즉 탑승객의 생존 유무에 대한 열이 없음
  - 예측에 해당하는 데이터이다

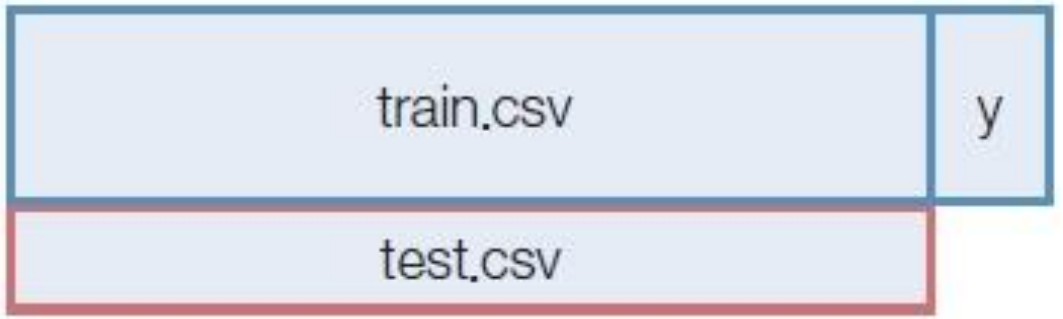


그림 6-9 train.csv와 test.csv의 관계

2.2 데이터 확인하기

- 다운로드한 데이터를 작업 폴더에 넣는다
  - 기본 예제 파일 경로는 'C:/source/ch06'
  - 코드 사용이 용이하도록 gender_submission.csv 파일은 삭제

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.2 데이터 확인하기

In [1]:	import pandas as pd import os import matplotlib.pyplot as plt import numpy as np import seaborn as sns sns.set(style="whitegrid", color_codes=True) DATA_DIR = 'c:/source/ch06/' os.listdir(DATA_DIR)
Out [1]:	['test.csv', 'train.csv']

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.2 데이터 확인하기

In [2]:	# test.csv과 test.csv를 가져온 후, 파일 순서 바꾸고 상대 경로 리스트 생성
	DATA_DIR = 'c:/source/ch06/'
	data_files = sorted([os.path.join(DATA_DIR, filename)
	for filename in os.listdir(DATA_DIR)], reverse=True)
	data_files

Out [2]:	['c:/source/ch06/train.csv', 'c:/source/ch06/test.csv']

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.2 데이터 확인하기

In [3]:

```
# (1) 데이터프레임을 각 파일에서 읽어온 후 df_list에 추가
df_list = []
for filename in data_files:
    df_list.append(pd.read_csv(filename))

# (2) 두 개의 데이터프레임을 하나로 통합
df = pd.concat(df_list, sort=False)

# (3) 인덱스 초기화
df = df.reset_index(drop=True)

# (4) 결과 출력
df.head(5)
```

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.2 데이터 확인하기

Out [3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0.0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.3 데이터 열 확인하기

표 6-1 타이타닉 데이터셋

변수명	의미	값 종류
Survived	생존 여부	0 = No, 1 = Yes
Pclass	티켓 클래스	1 = 1st, 2 = 2nd, 3 = 3rd
Sex	성별	
Age	나이	
SibSp	타이타닉 밖의 형제자매/부부의 수	
Parch	타이타닉 밖의 부모/자식의 수	
Ticket	티켓 번호	
Fare	티켓 가격	
Cabin	객실번호	
Embarked	승선항구	C = Cherbourg, Q = Queenstown, S = Southampton

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.3 데이터 열 확인하기

In [4]:

```
# (1) train.csv 데이터의 수
number_of_train_dataset = df.Survived.notnull().sum()
# (2) test.csv 데이터의 수
number_of_test_dataset = df.Survived.isnull().sum()
# (3) train.csv 데이터의 y 값 추출
y_true = df.pop("Survived")[:number_of_train_dataset]
```

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.4 데이터 노트 작성하기

- 데이터 노트 : 분석해야 하는 데이터에 대한 여러 가지 아이디어를 정리하는 노트
  - 각 데이터의 현재 데이터 타입 올바르게 정의
    - 숫자로 표시되어 있지만 범주형 데이터로 변형이 필요한 경우 등

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.4 데이터 노트 작성하기

표 6-2 데이터 노트의 예시

변수명	의미	데이터 타입	아이디어
Survived	생존 여부	범주형	Y 데이터
Pclass	티켓 클래스	범주형	
Sex	성별	범주형	
Age	나이	범주형	생존여부에 나이가 영향을 줄까?
SibSp	타이타닉 밖의 형제자매/부부의 수	연속형(int)	
Parch	타이타닉 밖의 부모/자식의 수	연속형(int)	
Ticket	티켓 번호	범주형	
Fare	티켓 가격	연속형(int)	티켓 가격과 pclass와 관련있지 않나?
Cabin	객실번호	범주형	
Embarked	승선항구	범주형	승선항구와 생존률은 영향이 있을까?

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.4 데이터 노트 작성하기

- 데이터의 모양을 확인할 때 T 함수 사용
  - transpose 함수는 데이터를 가로로 한 줄씩 보여줘 안에 있는 값들을 확인하기 좋음

In [5]:
```
df.head(2).T
```

Out [5]:

	0	1
PassengerId	1	2
Pclass	3	1
Name	Braund, Mr. Owen Harris	Cumings, Mrs. John Bradley (Florence Briggs Th...
Sex	male	female
Age	22.0	38.0
SibSp	1	1
Parch	0	0
Ticket	A/5 21171	PC 17599
Fare	7.25	71.2833
Cabin	NaN	C85
Embarked	S	C

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.5 결측치 확인하기

- 열별로 결측치 비율을 확인하여 전략을 세움

In [6]:

```
# (1) 데이터를 소수점 두 번째 자리까지 출력
pd.options.display.float_format = '{:.2f}'.format

# (2) 결측치 값의 합을 데이터의 개수로 나눠 비율로 출력
df.isnull().sum() / len(df) * 100
```

Out [6]:

```
PassengerId   0.00
Pclass        0.00
Name          0.00
Sex           0.00
Age          20.09
SibSp         0.00
Parch         0.00
Ticket        0.00
Fare          0.08
Cabin        77.46
Embarked      0.15
dtype: float64
```

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.5 결측치 확인하기

- 데이터를 삭제할지 전략적인 의사결정
- 결측치를 채우는 방법을 결정

In [7]:	df[df["Age"].notnull()].groupby(["Sex"])["Age"].mean()
Out [7]:	Sex female 28.69 male 30.59 Name: Age, dtype: float64

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.5 결측치 확인하기

- 데이터를 삭제할지 전략적인 의사결정
- 결측치를 채우는 방법을 결정

In [8]:	df[df["Age"].notnull()].groupby(["Pclass"])["Age"].mean()
Out [8]:	Pclass 1 39.16 2 29.51 3 24.82 Name: Age, dtype: float64

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.5 결측치 확인하기

In [9]:
```
df["Age"].fillna( df.groupby("Pclass")["Age"].transform("mean"), inplace=True)

df.isnull().sum() / len(df) * 100
```

Out [9]:
```
PassengerId    0.00
Pclass         0.00
Name           0.00
Sex            0.00
Age            0.00
SibSp          0.00
Parch          0.00
Ticket         0.00
Fare           0.08
Cabin         77.46
Embarked       0.15
dtype: float64
```

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.5 결측치 확인하기

- 데이터의 특성을 더 잘 나타내는 값으로 채워넣음, 분석결과 S가 많음

In [10]:
```
df.loc[61,"Embarked"] = "S"
df.loc[829,"Embarked"] = "S"
```

Passenger	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
62	1	1	Icard, Mis	female	38	0	0	113572	80	B28	
830	1	1	Stone, Mr	female	62	0	0	113572	80	B28	

(a) train.csv file에서 Null인 경우

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
61	62	1	Icard, Miss. Amelie	female	38.00	0	0	113572	80.00	B28	S
829	830	1	Stone, Mrs. George Nelson (Martha Evelyn)	female	62.00	0	0	113572	80.00	B28	S

(b) df 의 dataframe 에서 "S"로 채워진 결과

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.6 범주형 데이터 처리 : 원핫인코딩

- 데이터 형태에 따라 처리 방법 결정
- df.**info**() 함수 : 열별로 데이터 타입을 확인
  - 열별로 문자열 리스트 타입으로 정리

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.6 범주형 데이터 처리 : 원핫인코딩

In [11]:

```
df.info()
```

Out [11]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 11 columns):
 #  Column       Non-Null  Count     Dtype
--- ------------ --------- --------- -----
 0  PassengerId  1309      non-null  int64
 1  Pclass       1309      non-null  int64
 2  Name         1309      non-null  object
 3  Sex          1309      non-null  object
 4  Age          1309      non-null  float64
 5  SibSp        1309      non-null  int64
 6  Parch        1309      non-null  int64
 7  Ticket       1309      non-null  object
 8  Fare         1308      non-null  float64
 9  Cabin        295       non-null  object
10  Embarked     1309      non-null  object
dtypes: float64(2), int64(4), object(5)
memory usage: 112.6+ KB
```

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.6 범주형 데이터 처리 : 원핫인코딩

- 데이터의 타입을 정리

In [12]:
```
object_columns = ["PassengerId", "Pclass", "Name", "Sex", "Ticket", "Cabin",
"Embarked"]
numeric_columns = ["Age", "SibSp", "Parch", "Fare"]

for col_name in object_columns:
    df[col_name] = df[col_name].astype(object)
for col_name in numeric_columns:
    df[col_name] = df[col_name].astype(float)

df["Parch"] = df["Parch"].astype(int)
df["SibSp"] = df["SibSp"].astype(int)
```

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.6 범주형 데이터 처리 : 원핫인코딩

- 데이터를 원핫인코딩으로 처리
- 데이터 유형이 작은 Sex, Pclass, Embarked를 원핫인코딩으로 처리함
- 원본데이터와 변형된 데이터를 병합(merge)하는 함수를 만듦

In [13]:
```
def merge_and_get(ldf, rdf, on, how="inner", index=None):
    if index is True:
        return pd.merge(ldf,rdf, how=how, left_index=True, right_index=True)
    else:
        return pd.merge(ldf,rdf, how=how, on=on)
```

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.6 범주형 데이터 처리 : 원핫인코딩

- 데이터를 원핫인코딩으로 처리
- get_dummies를 활용하여 데이터의 원핫인코딩 벡터를 생성한 후 원본 데이터와 통합함

In [14]:
```
one_hot_df = merge_and_get(
    df, pd.get_dummies(df["Sex"], prefix="Sex"), on=None, index=True)
one_hot_df = merge_and_get(
    one_hot_df, pd.get_dummies(
        df["Pclass"], prefix="Pclass"), on=None, index=True)
one_hot_df = merge_and_get(
    one_hot_df, pd.get_dummies(
        df["Embarked"], prefix="Embarked"), on=None, index=True)
```

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.7 데이터 시각화 진행하기

- y 값과 각 범주형 타입 간에 어떤 관계가 있는지를 확인
- 열별로 y_true 데이터와 합쳐서 비교 그래프로 나타내어 각 열이 생존 여부에 영향을 주는지 **시각적으로 확인하는 것은 매우 중요한 데이터 분석 기법 중의 하나**임
  - 데이터 유형별로 y_true 데이터의 분포 변화가 있는가

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.7 데이터 시각화 진행하기

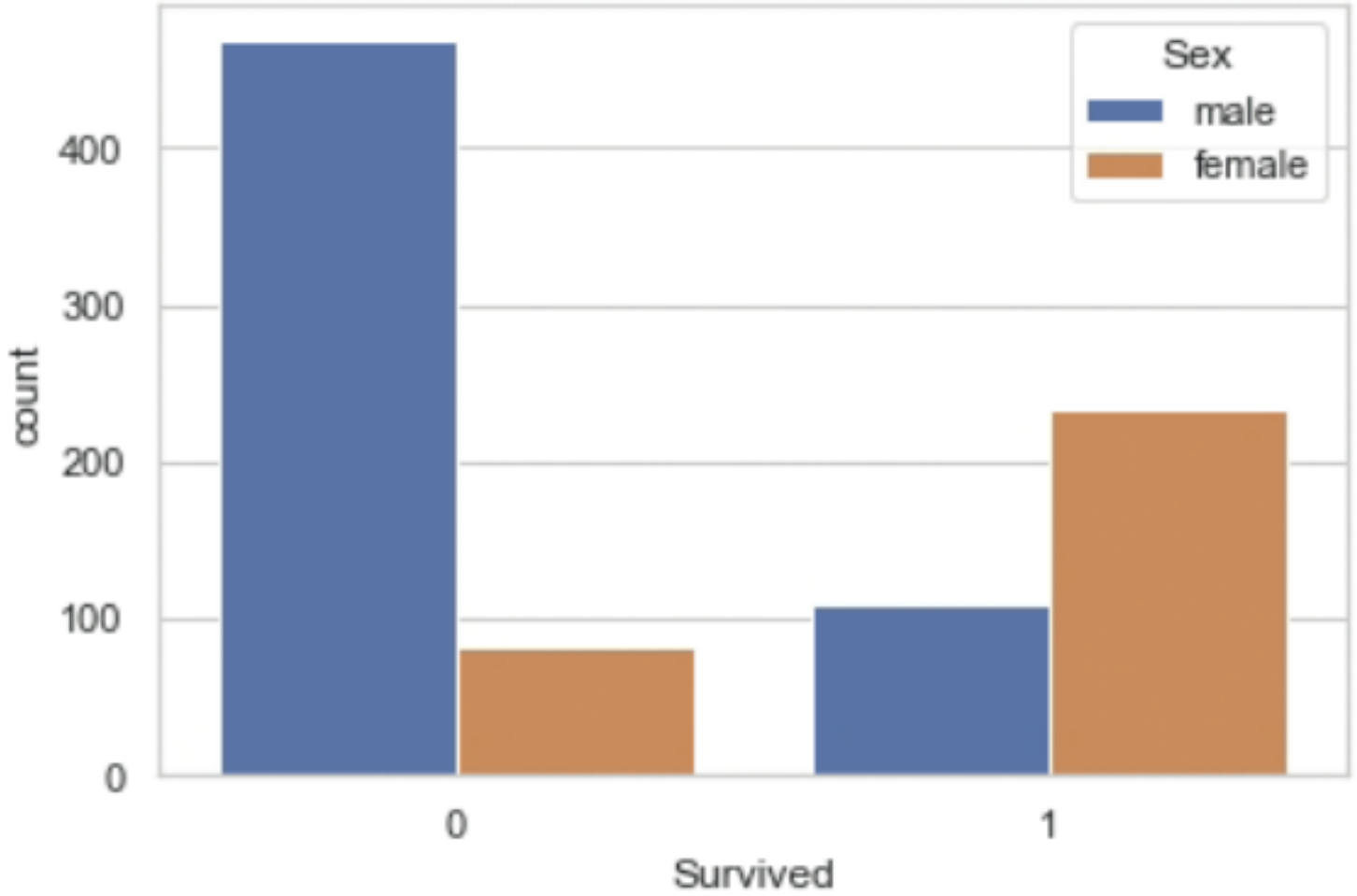
In [15]:
```
temp_columns = ["Sex", "Pclass", "Embarked"]
for col_name in temp_columns:
    temp_df = pd.merge(one_hot_df[col_name], y_true, left_index=True,
right_index=True)
    sns.countplot(x="Survived", hue=col_name, data=temp_df)
    plt.show()
```

03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.7 데이터 시각화 진행하기

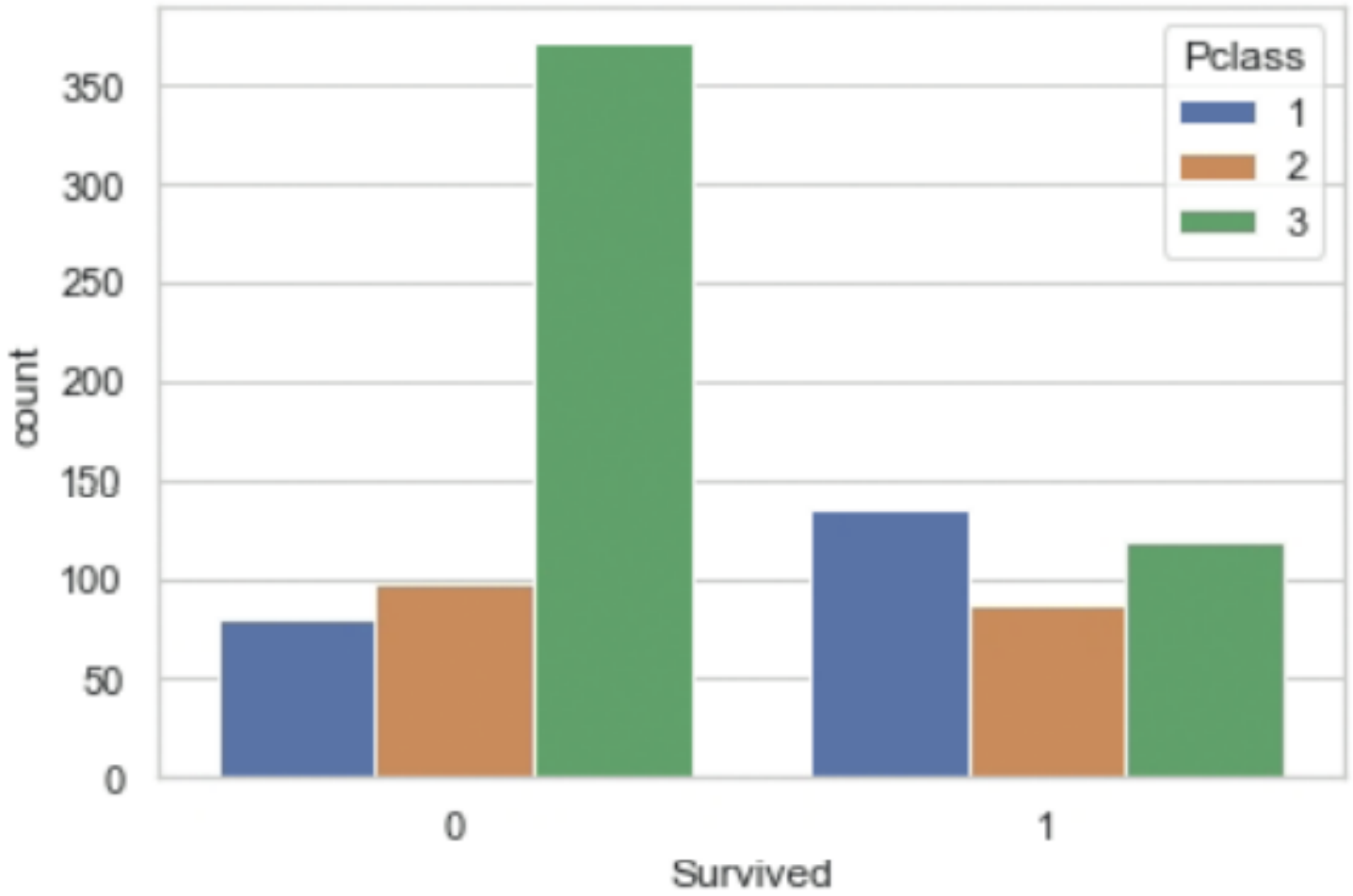
Out [15]:



2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.7 데이터 시각화 진행하기

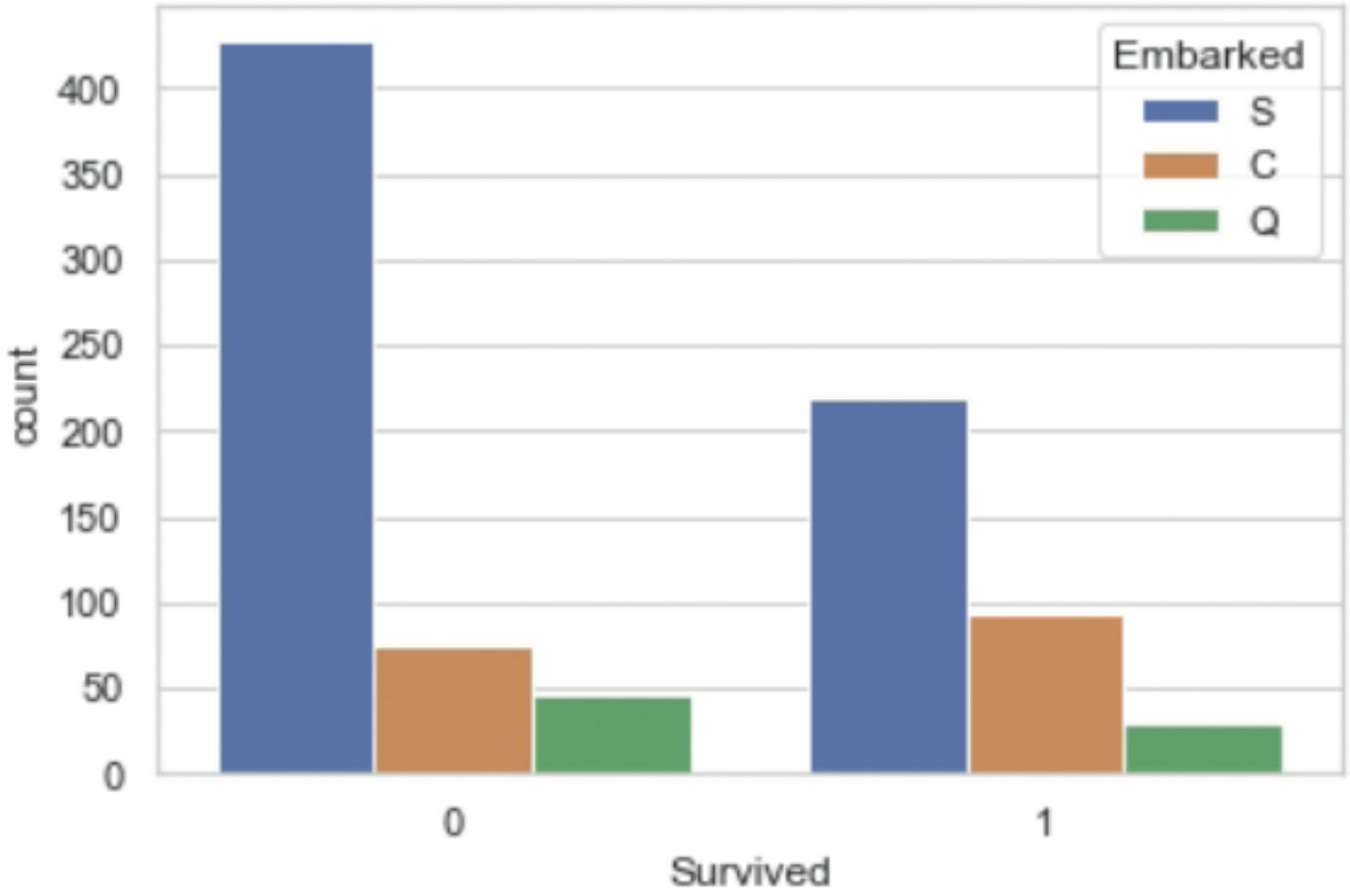
Out [15]:



2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.7 데이터 시각화 진행하기

Out [15]:



03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.7 데이터 시각화 진행하기

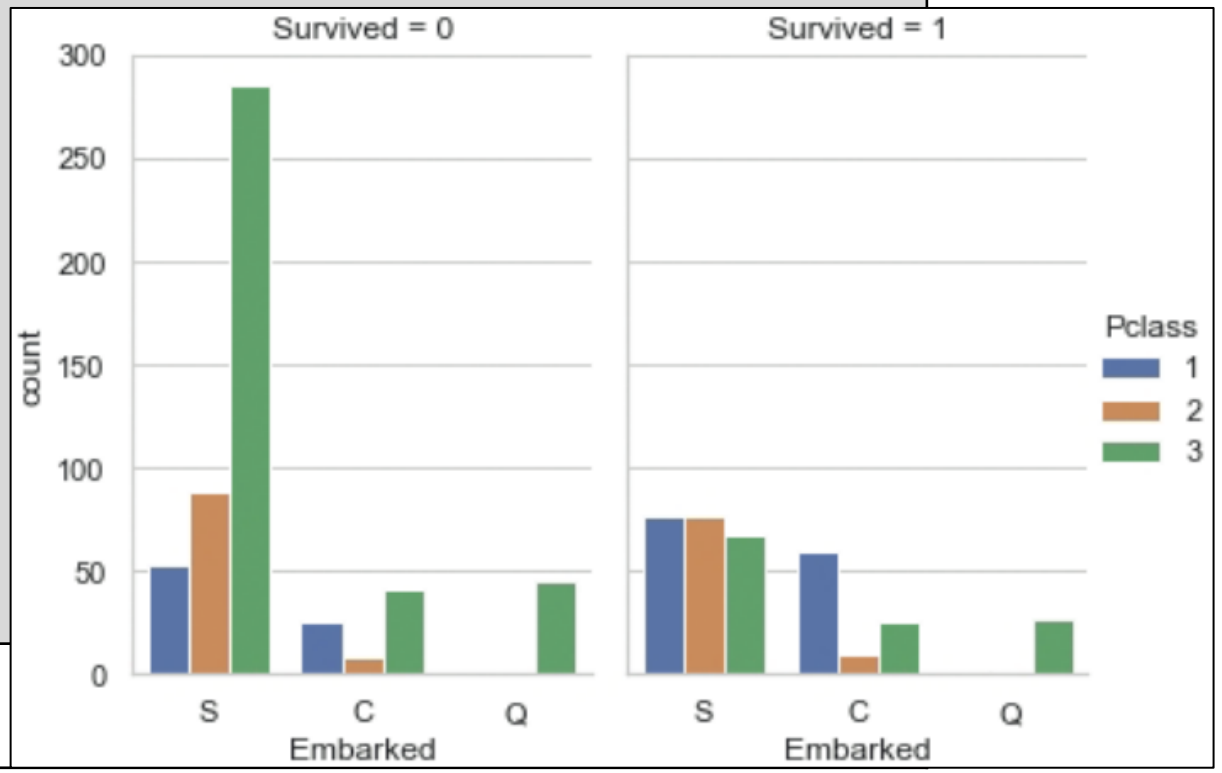
- 범주형 데이터 간 상관관계 분석
- 예: Sex와 Pclass 사이의 관계는 어떠한가?

In [16]:

```
temp_df = pd.merge(one_hot_df[temp_columns],
                   y_true, left_index=True,
                   right_index=True)

g = sns.catplot(x="Embarked",
                hue="Pclass",
                col="Survived",
                data=temp_df,
                kind="count",
                height=4, aspect=.7);
```

Out [16]:



03 데이터 전처리의 실습

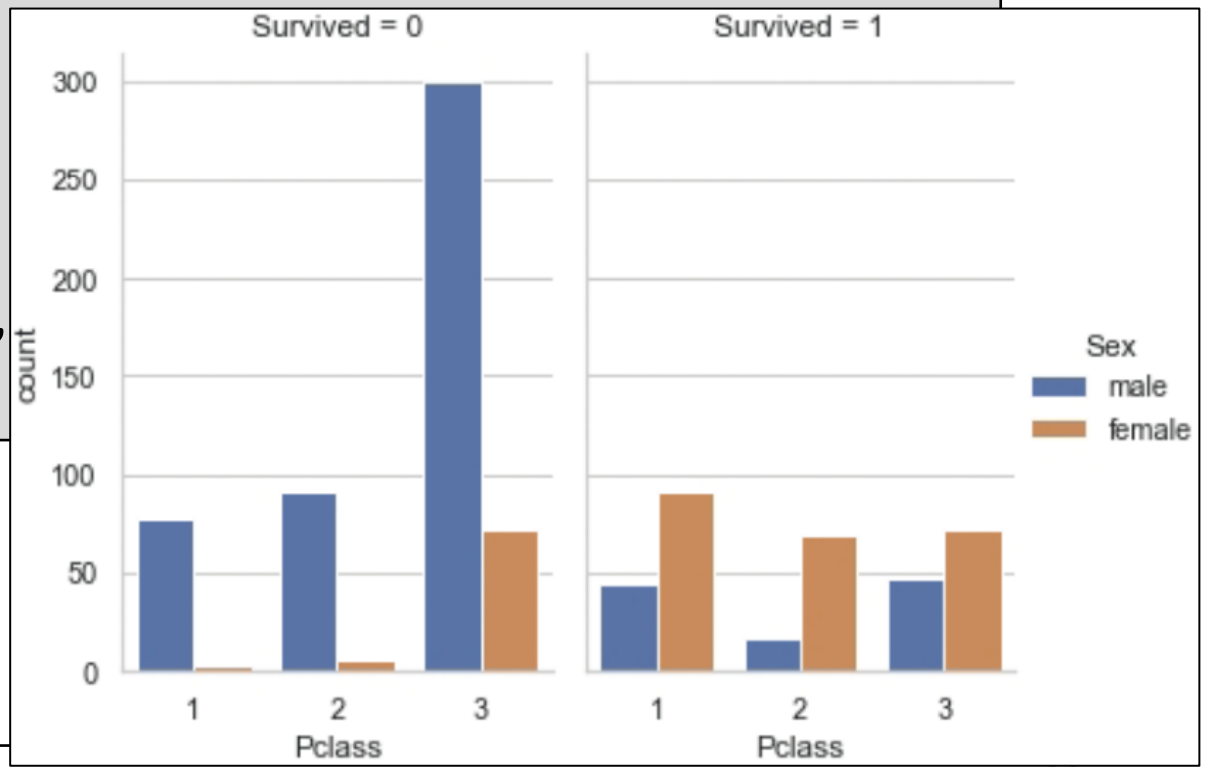
2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.7 데이터 시각화 진행하기

In [17]:
```
temp_df = pd.merge(
    one_hot_df[temp_columns],
    y_true, left_index=True,
    right_index=True)

g = sns.catplot(x="Pclass",
                hue="Sex", col="Survived",
                data=temp_df, kind="count",
                height=4, aspect=.7)
```

Out [17]:



03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

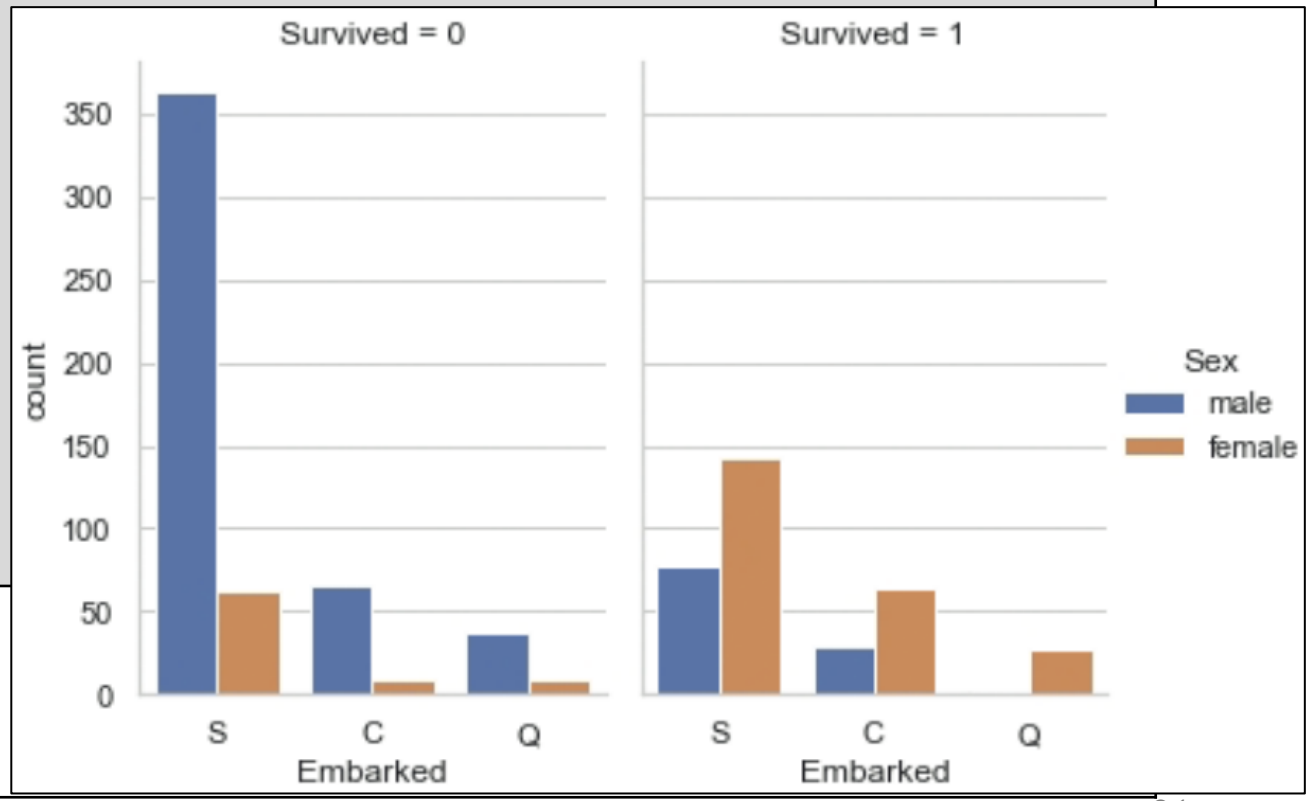
2.7 데이터 시각화 진행하기

In [18]:

```
temp_df = pd.merge(
    one_hot_df[temp_columns],
    y_true, left_index=True,
    right_index=True)

g = sns.catplot(
    x="Embarked", hue="Sex",
    col="Survived",
    data=temp_df, kind="count",
    height=4, aspect=.7);
```

Out [18]:



03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.7 데이터 시각화 진행하기

- Heatmap 함수 : 상관계수(correlation) 데이터로 확인
  - **corr 함수로 상관계수 계산**

In [19]:

```
crosscheck_columns = [col_name for col_name in one_hot_df.columns.tolist()
   if col_name.split("_")[0] in temp_columns and "_" in col_name ] + ["Sex"]
# temp 열
temp_df = pd.merge(one_hot_df[crosscheck_columns], y_true, left_index=True,
right_index=True)
corr = temp_df.corr()
sns.set()
ax = sns.heatmap(corr, annot=True, linewidths=.5, cmap="YlGnBu")
```

*annot=True: 각 셀에 숫자를 입력, cmap: 색깔 지정

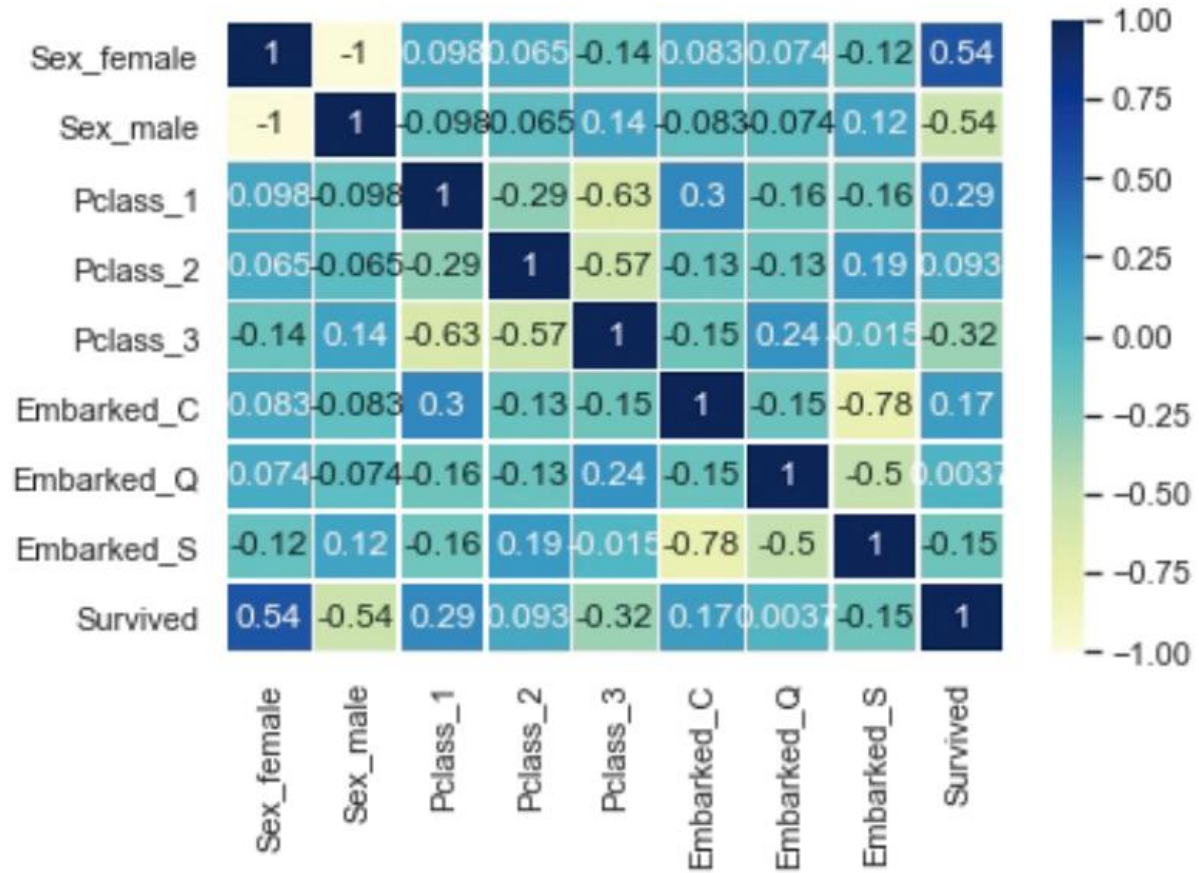
03 데이터 전처리의 실습

2. 데이터 전처리 실습하기 : 타이타닉 생존자 예측하기

2.7 데이터 시각화 진행하기

- Sex_female이 Survived에 가장 많은 영향을 줌

Out [19]:



Assignment

Assignment

- 다음 코드를 실행할 때 예상되는 결과값을 적으시오. (코드와 함께 결과물 제출할 것)

```
import numpy as np
import pandas as pd

raw_data = {'first_name': ['Jason', np.nan, 'Tina', 'Jake', 'Amy'],
            'last_name': ['Miller', np.nan, 'Ali', 'Milner', 'Cooze'],
            'age': [42, np.nan, 36, 24, 73],
            'sex': ['m', np.nan, 'f', 'm', 'f'],
            'preTestScore': [4, np.nan, np.nan, 2, 3],
            'postTestScore': [25, np.nan, np.nan, 62, 70]}
df = pd.DataFrame(raw_data, columns = [
                                       'first_name', 'last_name', 'age', 'sex',
                                       'preTestScore', 'postTestScore'])

df.isnull().sum()
```

Thank You !