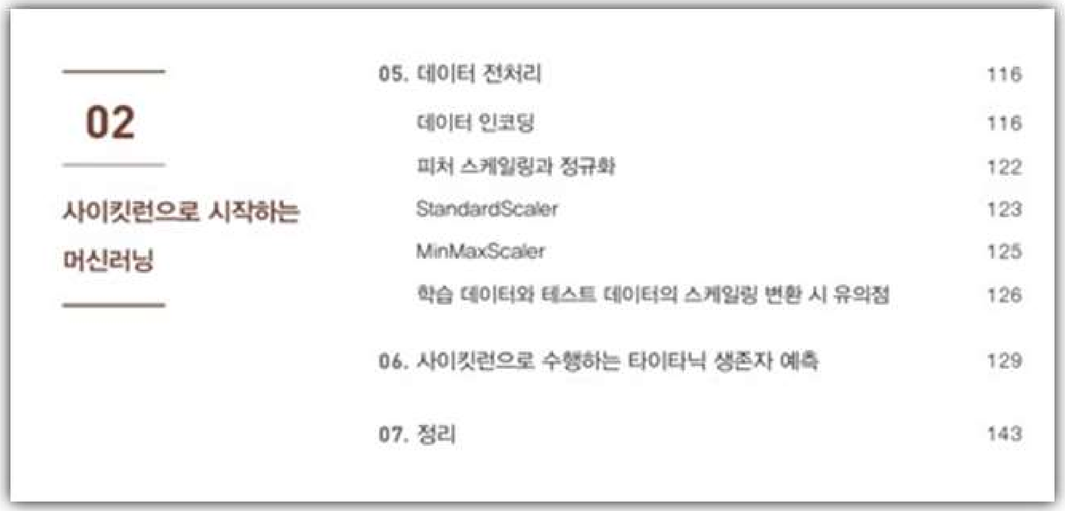
3rd Meet

Long Time No See

□ 오늘 공부할 것은



02

사이킷런으로 시작하는 머신러닝

Now, It's your turn ???

40min ~ 50min

[02-05] 데이터 전처리 (Data Preprocessing)

- 데이터 인코딩 (Encoding)
  . Label Encoding
  . One-Hot Encoding
- 피처 스케일링과 정규화 (Standardization and Normalization)
  . StandardScaler
  . MinMaxScaler
  . 학습 데이터와 테스트 데이터의 스케일링 변환 시 유의점

[02-06] 사이킷런으로 수행하는 타이타닉 생존자 예측

[02-07] 정리

[02-05]
데이터 전처리
(Data Preprocessing)

□ Data Preprocessing

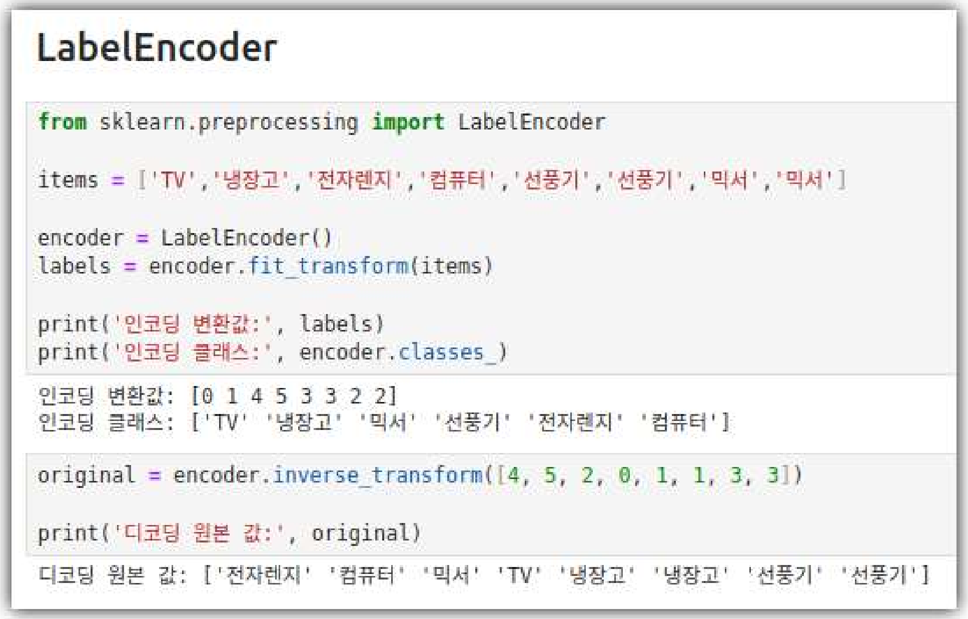
- 결손값 / 결측치 / NaN / Null → 버리거나 평균치 등을 이용한 대체
- 문자열 / String → 숫자형으로 변환(encoding)

  . 카테고리형 → 코드 값으로 대체

  . 텍스트형 → 피처 벡터화 (feature vectorization) 또는 삭제

□ Encoding - Label encoding

- Category → 코드형 숫자 값으로 변환
- But 숫자의 크기는?
  . 트리 구조에서는 별 문제가 없으나, 선형 회귀에서는 문제



LabelEncoder

```
from sklearn.preprocessing import LabelEncoder

items = ['TV','냉장고','전자렌지','컴퓨터','선풍기','선풍기','믹서','믹서']

encoder = LabelEncoder()
labels = encoder.fit_transform(items)

print('인코딩 변환값:', labels)
print('인코딩 클래스:', encoder.classes_)
```

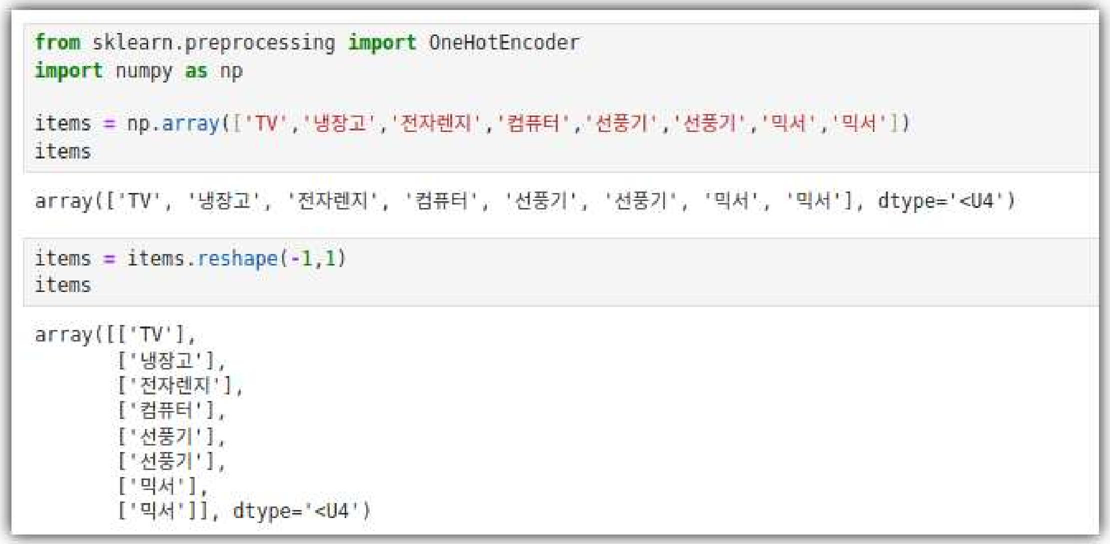
```
인코딩 변환값: [0 1 4 5 3 3 2 2]
인코딩 클래스: ['TV' '냉장고' '믹서' '선풍기' '전자렌지' '컴퓨터']
```

```
original = encoder.inverse_transform([4, 5, 2, 0, 1, 1, 3, 3])

print('디코딩 원본 값:', original)
```

```
디코딩 원본 값: ['전자렌지' '컴퓨터' '믹서' 'TV' '냉장고' '냉장고' '선풍기' '선풍기']
```

□ Encoding - One-Hot Encoding

- 입력 = numpy 행렬

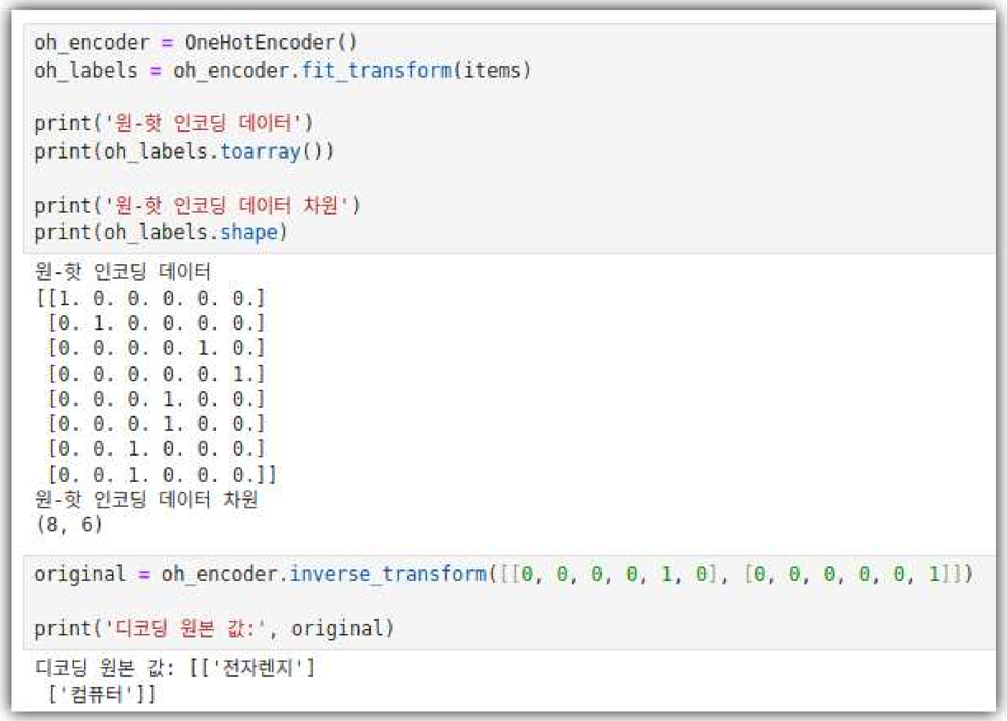


```
from sklearn.preprocessing import OneHotEncoder
import numpy as np

items = np.array(['TV','냉장고','전자렌지','컴퓨터','선풍기','선풍기','믹서','믹서'])
items
```

```
array(['TV', '냉장고', '전자렌지', '컴퓨터', '선풍기', '선풍기', '믹서', '믹서'], dtype='<U4')
```

```
items = items.reshape(-1,1)
items
```

```
array([['TV'],
       ['냉장고'],
       ['전자렌지'],
       ['컴퓨터'],
       ['선풍기'],
       ['선풍기'],
       ['믹서'],
       ['믹서']], dtype='<U4')
```



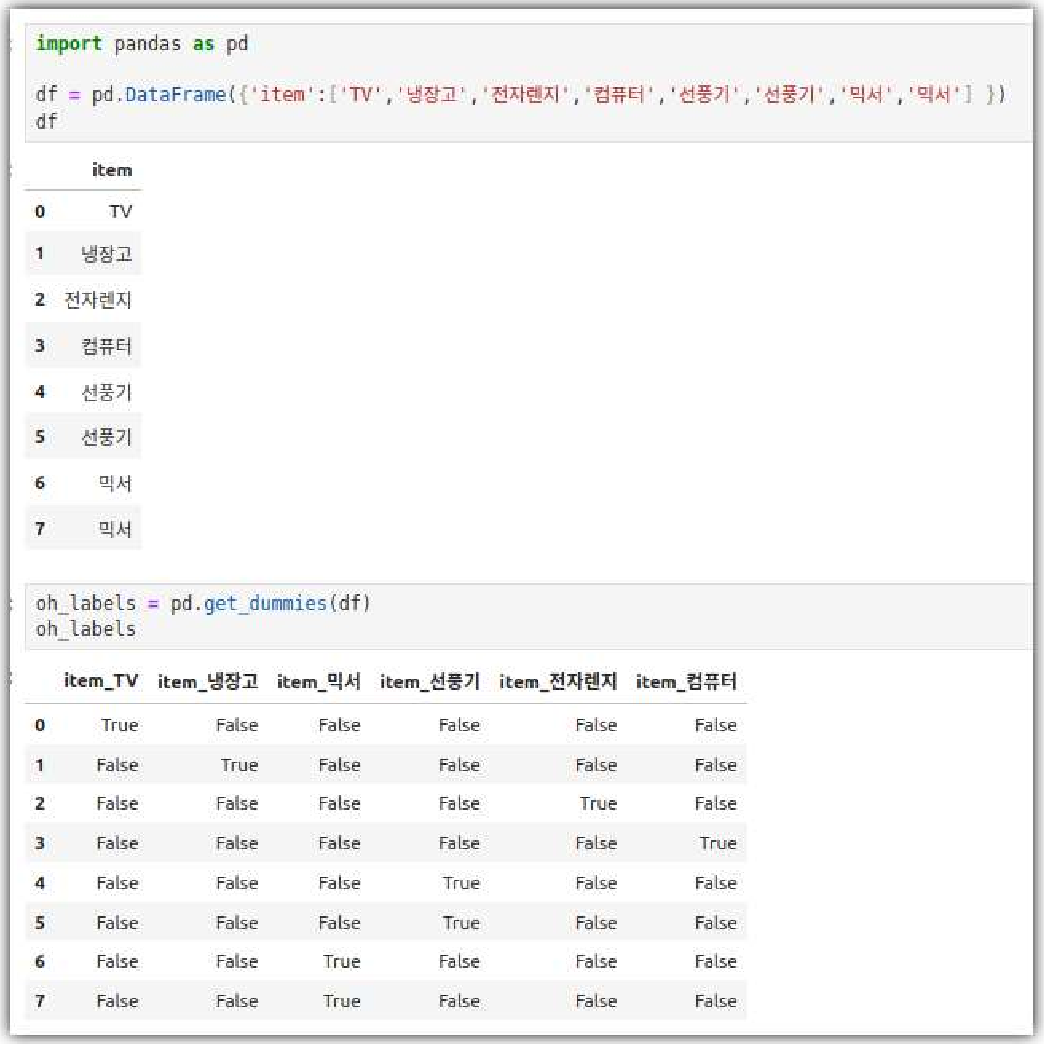
```
oh_encoder = OneHotEncoder()
oh_labels = oh_encoder.fit_transform(items)

print('원-핫 인코딩 데이터')
print(oh_labels.toarray())

print('원-핫 인코딩 데이터 차원')
print(oh_labels.shape)
```

```
원-핫 인코딩 데이터
[[1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0.]]
원-핫 인코딩 데이터 차원
(8, 6)
```

```
original = oh_encoder.inverse_transform([[0, 0, 0, 0, 1, 0], [0, 0, 0, 0, 0, 1]])

print('디코딩 원본 값:', original)
```

```
디코딩 원본 값: [['전자렌지']
 ['컴퓨터']]
```

□ Encoding - One-Hot Encoding



```
import pandas as pd

df = pd.DataFrame({'item':['TV','냉장고','전자렌지','컴퓨터','선풍기','선풍기','믹서','믹서'] })
df
```

	item
0	TV
1	냉장고
2	전자렌지
3	컴퓨터
4	선풍기
5	선풍기
6	믹서
7	믹서

```
oh_labels = pd.get_dummies(df)
oh_labels
```

	item_TV	item_냉장고	item_믹서	item_선풍기	item_전자렌지	item_컴퓨터
0	True	False	False	False	False	False
1	False	True	False	False	False	False
2	False	False	False	False	True	False
3	False	False	False	False	False	True
4	False	False	False	True	False	False
5	False	False	False	True	False	False
6	False	False	True	False	False	False
7	False	False	True	False	False	False

□ Feature Scaling - Standardization (표준화)

- 평균 = 0, 분산 = 1, Gaussian 분포

$$x_i\_new = \frac{x_i - mean(x)}{stdev(x)}$$

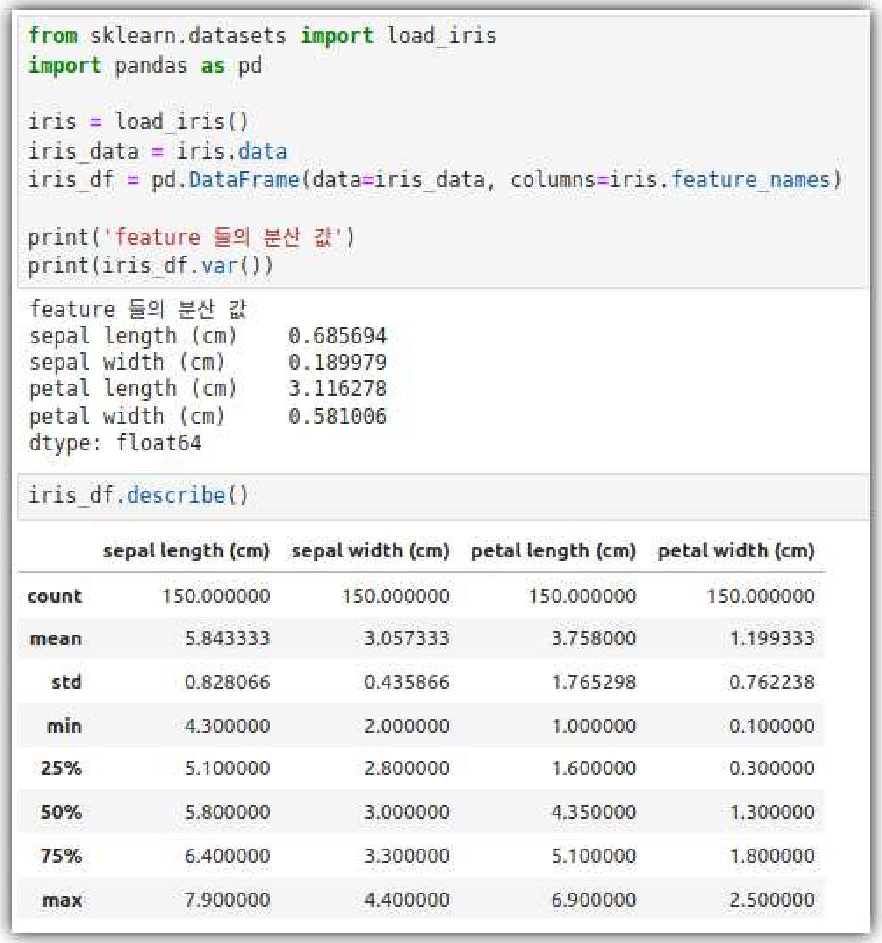
- Gaussian 분포 기반 (in 사이킷런)

  . SVM (Support Vector Machine)

  . Linear Regression

  . Logistic Regression

□ Feature Scaling - Standardization (표준화) - StandardScaler

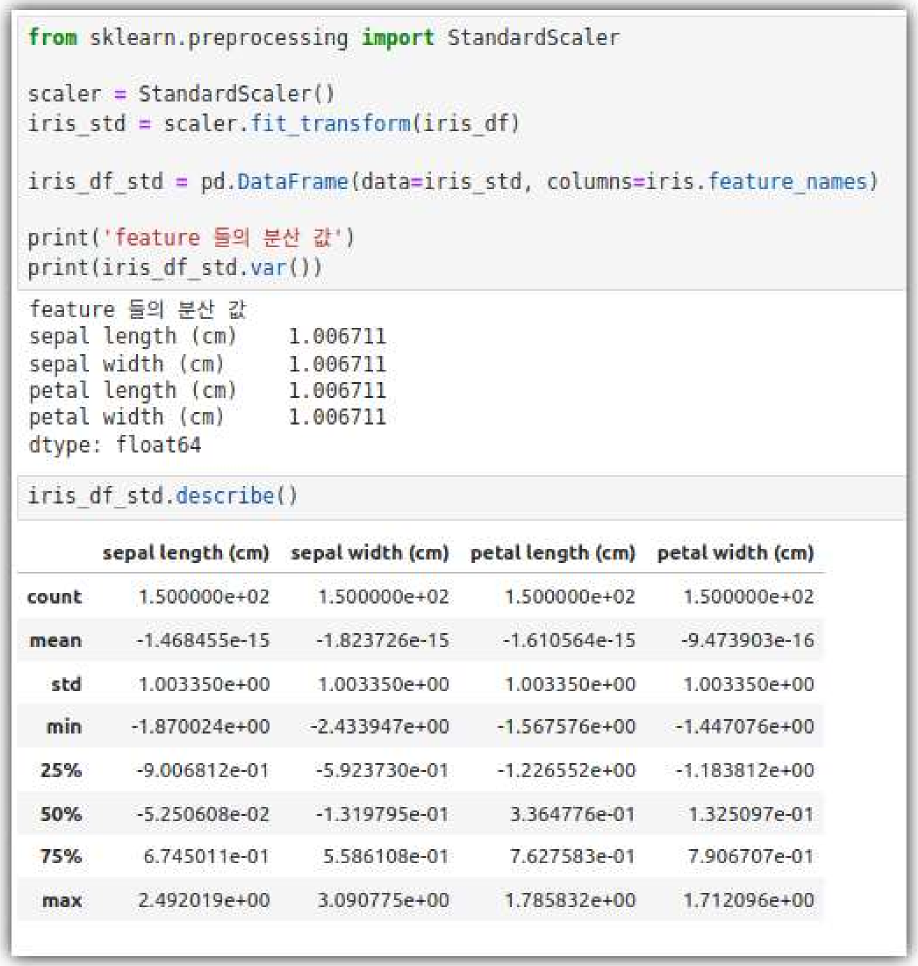


```python
from sklearn.datasets import load_iris
import pandas as pd

iris = load_iris()
iris_data = iris.data
iris_df = pd.DataFrame(data=iris_data, columns=iris.feature_names)

print('feature 들의 분산 값')
print(iris_df.var())
```

```
feature 들의 분산 값
sepal length (cm)    0.685694
sepal width (cm)     0.189979
petal length (cm)    3.116278
petal width (cm)     0.581006
dtype: float64
```

```python
iris_df.describe()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000



```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
iris_std = scaler.fit_transform(iris_df)

iris_df_std = pd.DataFrame(data=iris_std, columns=iris.feature_names)

print('feature 들의 분산 값')
print(iris_df_std.var())
```

```
feature 들의 분산 값
sepal length (cm)    1.006711
sepal width (cm)     1.006711
petal length (cm)    1.006711
petal width (cm)     1.006711
dtype: float64
```

```python
iris_df_std.describe()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	1.500000e+02	1.500000e+02	1.500000e+02	1.500000e+02
mean	-1.468455e-15	-1.823726e-15	-1.610564e-15	-9.473903e-16
std	1.003350e+00	1.003350e+00	1.003350e+00	1.003350e+00
min	-1.870024e+00	-2.433947e+00	-1.567576e+00	-1.447076e+00
25%	-9.006812e-01	-5.923730e-01	-1.226552e+00	-1.183812e+00
50%	-5.250608e-02	-1.319795e-01	3.364776e-01	1.325097e-01
75%	6.745011e-01	5.586108e-01	7.627583e-01	7.906707e-01
max	2.492019e+00	3.090775e+00	1.785832e+00	1.712096e+00

□ Feature Scaling - Normalization (정규화)

- 서로 다른 피처의 크기를 통일하기 위해 크기 변환

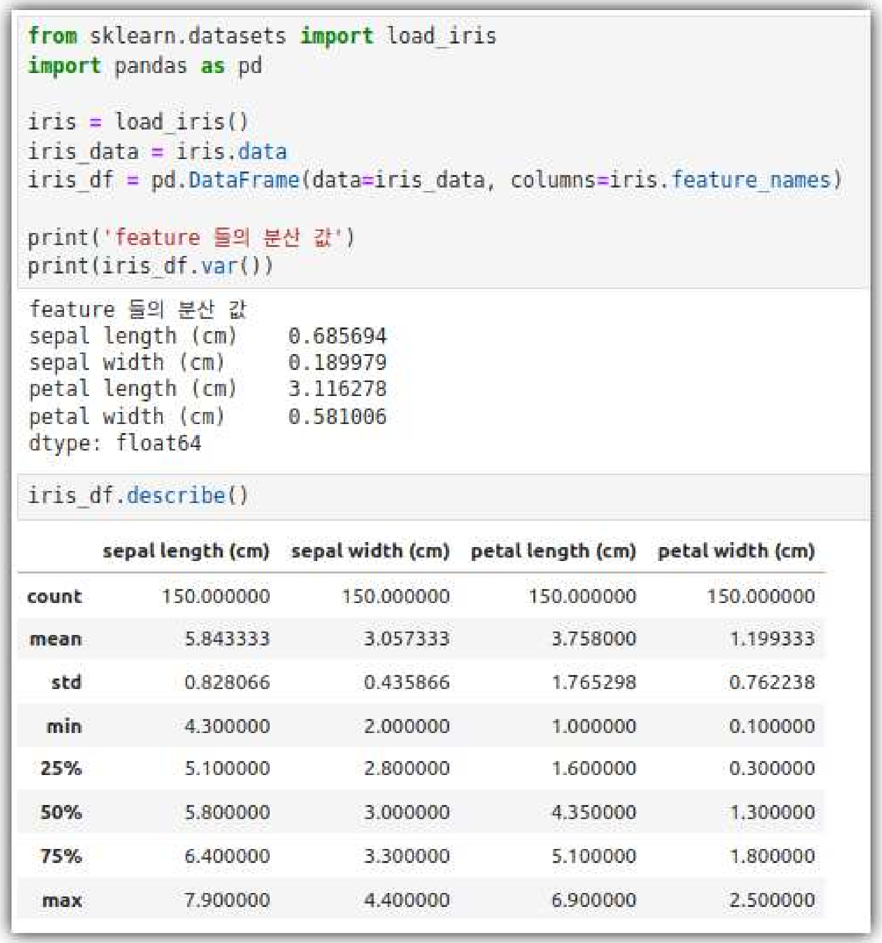
$$x_i\_new = \frac{x_i - min(x)}{\max(x) - \min(x)}$$

- '사이킷런'에서는 선형대수의 Normalization 방식 (= Vector Normalization)

$$x_i\_new = \frac{x_i}{\sqrt{x_i^2 + y_i^2 + z_i^2}}$$

□ Feature Scaling - Normalization (정규화) - MinMaxScaler

- 데이터의 값을 0과 1 사이의 범위로 변환 (음수가 있다면 -1에서 1 사이)

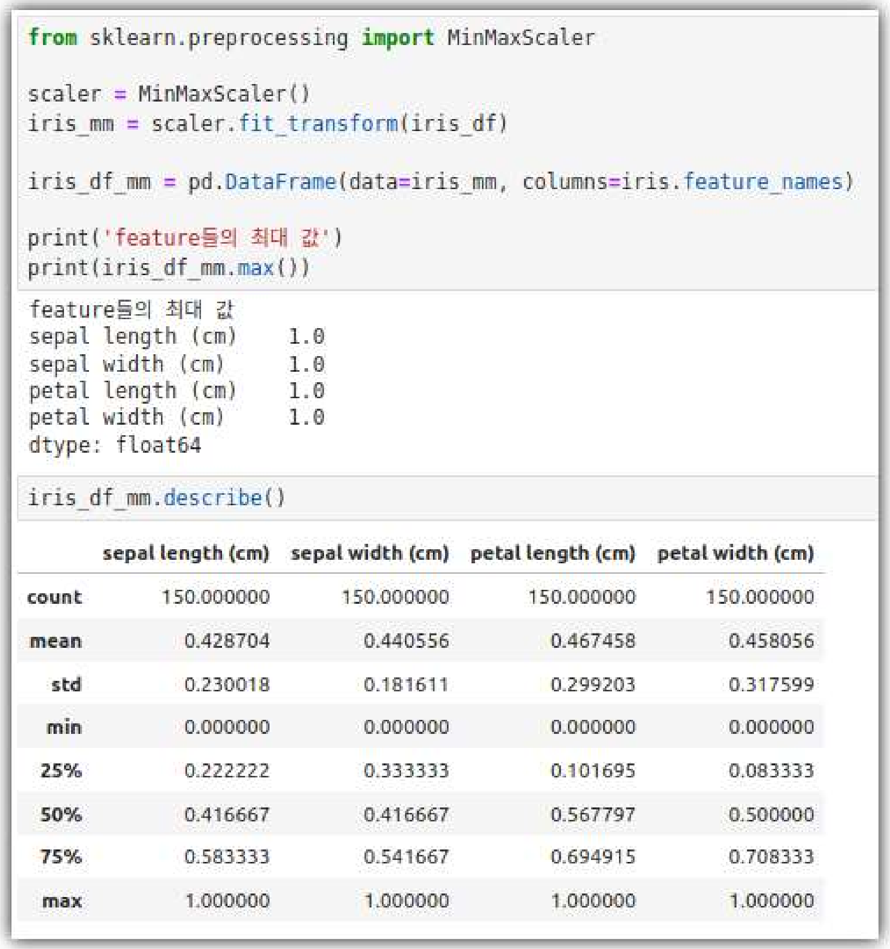


```
from sklearn.datasets import load_iris
import pandas as pd

iris = load_iris()
iris_data = iris.data
iris_df = pd.DataFrame(data=iris_data, columns=iris.feature_names)

print('feature 들의 분산 값')
print(iris_df.var())
```

```
feature 들의 분산 값
sepal length (cm)    0.685694
sepal width (cm)     0.189979
petal length (cm)    3.116278
petal width (cm)     0.581006
dtype: float64
```

```
iris_df.describe()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000



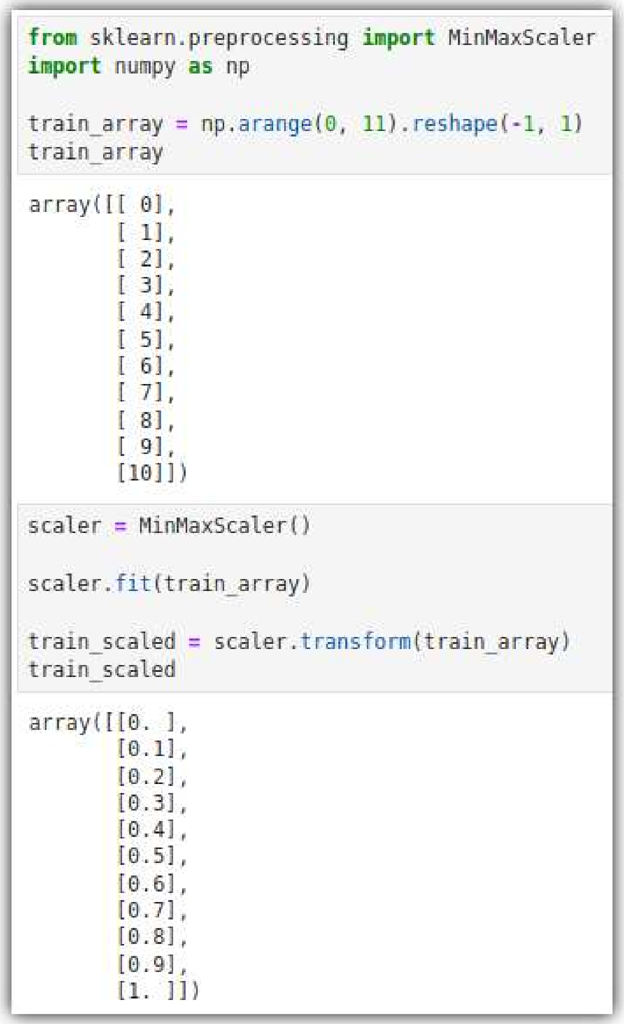
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
iris_mm = scaler.fit_transform(iris_df)

iris_df_mm = pd.DataFrame(data=iris_mm, columns=iris.feature_names)

print('feature들의 최대 값')
print(iris_df_mm.max())
```

```
feature들의 최대 값
sepal length (cm)    1.0
sepal width (cm)     1.0
petal length (cm)    1.0
petal width (cm)     1.0
dtype: float64
```

```
iris_df_mm.describe()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	0.428704	0.440556	0.467458	0.458056
std	0.230018	0.181611	0.299203	0.317599
min	0.000000	0.000000	0.000000	0.000000
25%	0.222222	0.333333	0.101695	0.083333
50%	0.416667	0.416667	0.567797	0.500000
75%	0.583333	0.541667	0.694915	0.708333
max	1.000000	1.000000	1.000000	1.000000

□ fit() + transform()

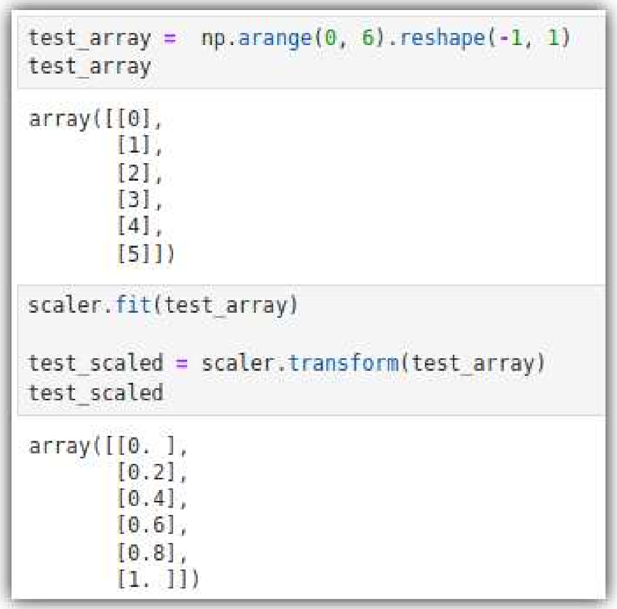


```
from sklearn.preprocessing import MinMaxScaler
import numpy as np

train_array = np.arange(0, 11).reshape(-1, 1)
train_array
```

```
array([[ 0],
       [ 1],
       [ 2],
       [ 3],
       [ 4],
       [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
       [10]])
```

```
scaler = MinMaxScaler()

scaler.fit(train_array)

train_scaled = scaler.transform(train_array)
train_scaled
```

```
array([[0. ],
       [0.1],
       [0.2],
       [0.3],
       [0.4],
       [0.5],
       [0.6],
       [0.7],
       [0.8],
       [0.9],
       [1. ]])
```

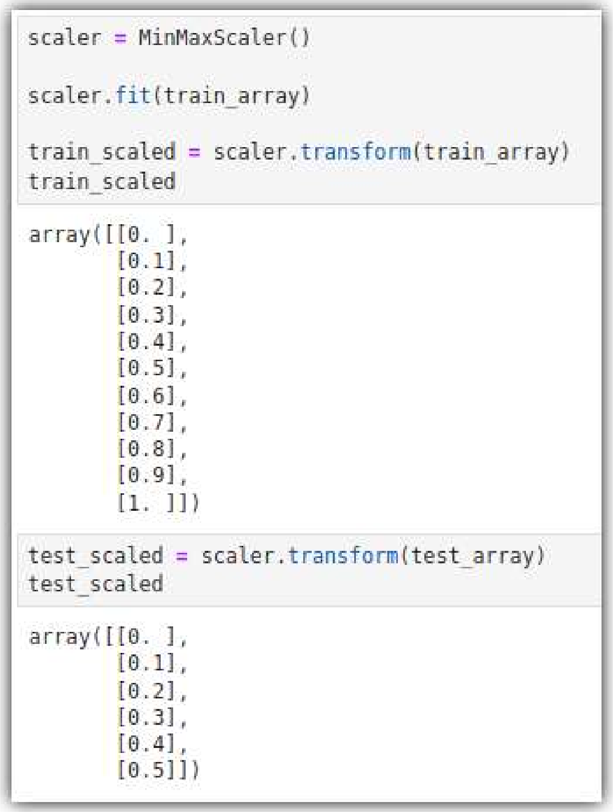


```
test_array =  np.arange(0, 6).reshape(-1, 1)
test_array
```

```
array([[0],
       [1],
       [2],
       [3],
       [4],
       [5]])
```

```
scaler.fit(test_array)

test_scaled = scaler.transform(test_array)
test_scaled
```

```
array([[0. ],
       [0.2],
       [0.4],
       [0.6],
       [0.8],
       [1. ]])
```



```
scaler = MinMaxScaler()

scaler.fit(train_array)

train_scaled = scaler.transform(train_array)
train_scaled
```

```
array([[0. ],
       [0.1],
       [0.2],
       [0.3],
       [0.4],
       [0.5],
       [0.6],
       [0.7],
       [0.8],
       [0.9],
       [1. ]])
```

```
test_scaled = scaler.transform(test_array)
test_scaled
```

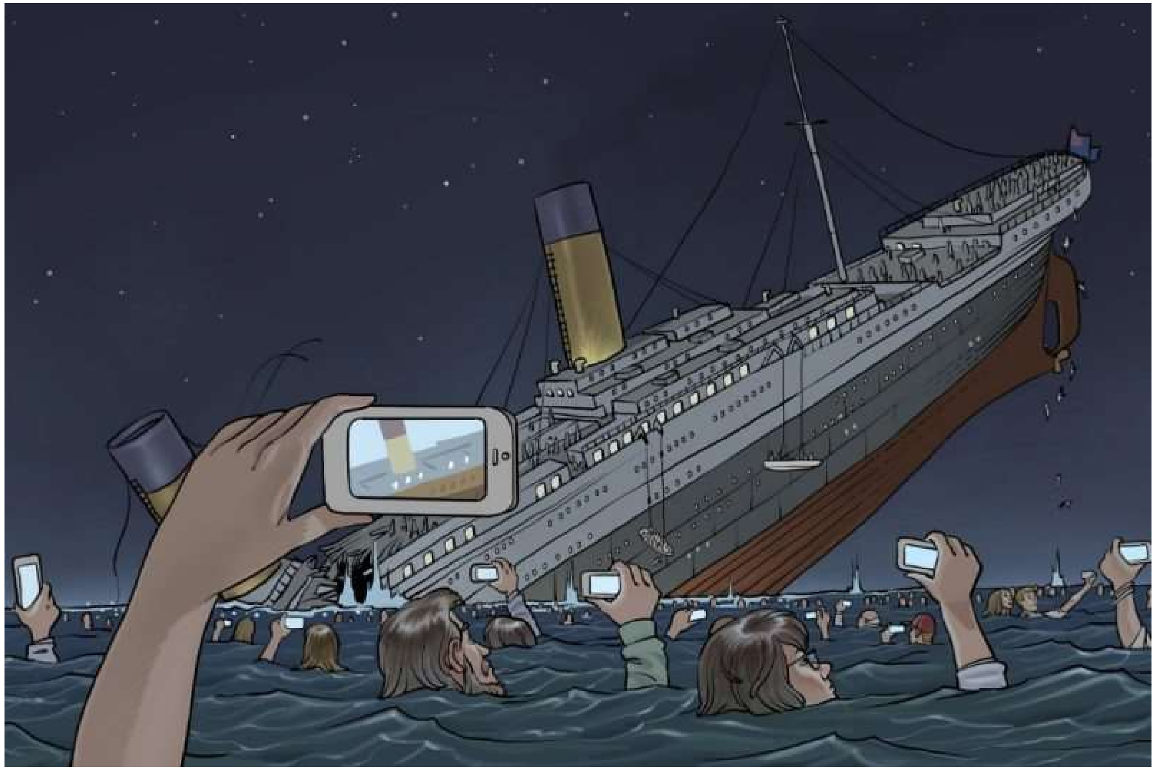
```
array([[0. ],
       [0.1],
       [0.2],
       [0.3],
       [0.4],
       [0.5]])
```

- 같은 fit()을 이용한 transform() 중요 !!!

[02-06]
사이킷런으로 수행하는 타이타닉 생존자 예측

RMS Titanic

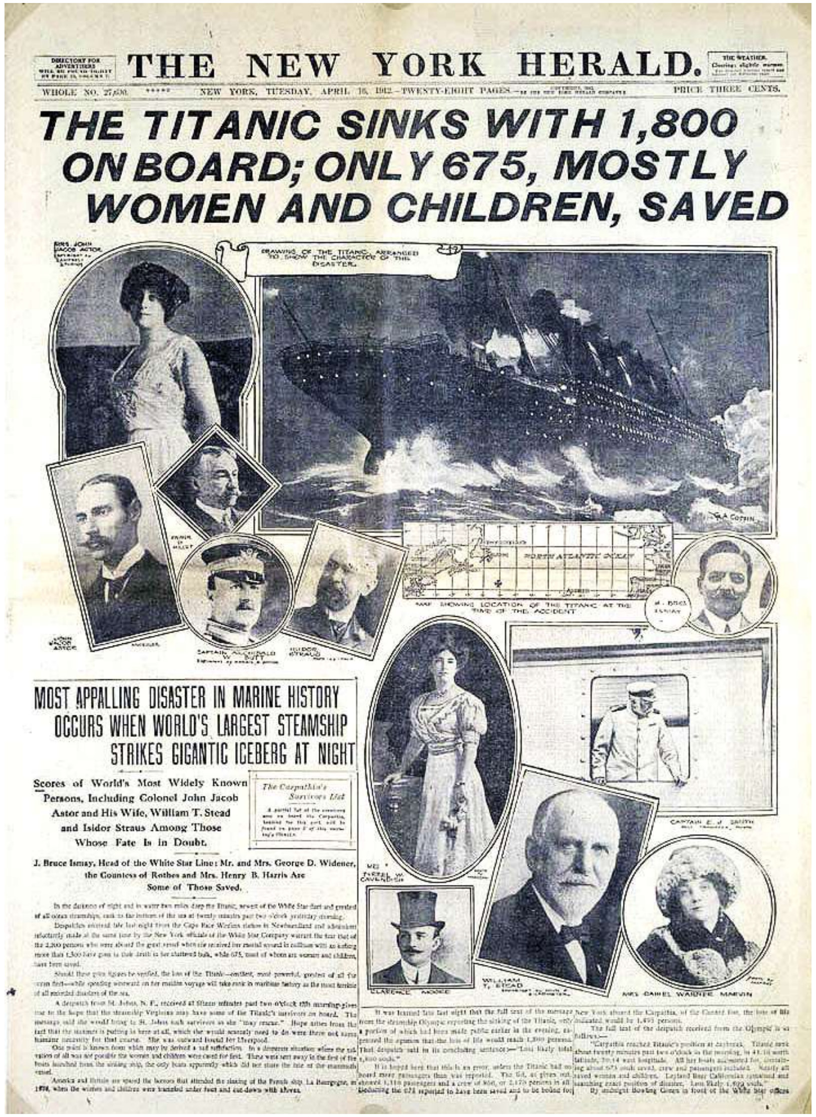
- if the titanic sank today...



https://www.reddit.com/r/Duklock/comments/bexm2x/if_the_titanic_sank_today/

Sinking

- 1912년 4월 14일 밤 침몰. 총 2224명 中 710명 구조, 1514명 사망



DIRECTORY FOR ADVENTURERS [illegible]

THE NEW YORK HERALD.

THE WEATHER. Clearing; slightly warmer. [illegible]

WHOLE NO. 27,630. ***** NEW YORK, TUESDAY, APRIL 16, 1912.—TWENTY-EIGHT PAGES.— [illegible] PRICE THREE CENTS.

THE TITANIC SINKS WITH 1,800 ON BOARD; ONLY 675, MOSTLY WOMEN AND CHILDREN, SAVED

MRS. JOHN JACOB ASTOR
DRAWING OF THE TITANIC, ARRANGED TO SHOW THE CHARACTER OF THE DISASTER.
MAP SHOWING LOCATION OF THE TITANIC AT THE TIME OF THE ACCIDENT
NORTH ATLANTIC OCEAN
JOHN JACOB ASTOR
CAPTAIN ARCHIBALD W. BUTT
ISIDOR STRAUS
CAPTAIN E. J. SMITH
MRS. TYRELL W. CAVENDISH
CLARENCE MOORE
WILLIAM T. STEAD
MRS. DANIEL WARNER MARVIN

MOST APPALLING DISASTER IN MARINE HISTORY OCCURS WHEN WORLD'S LARGEST STEAMSHIP STRIKES GIGANTIC ICEBERG AT NIGHT

Scores of World's Most Widely Known Persons, Including Colonel John Jacob Astor and His Wife, William T. Stead and Isidor Straus Among Those Whose Fate Is in Doubt.

The Carpathia's Survivors' List

A partial list of the survivors [illegible]

J. Bruce Ismay, Head of the White Star Line; Mr. and Mrs. George D. Widener, the Countess of Rothes and Mrs. Henry B. Harris Are Some of Those Saved.

In the darkness of night and in water two miles deep the Titanic, newest of the White Star fleet and greatest of all ocean steamships, sank to the bottom of the sea at twenty minutes past two o'clock yesterday morning.

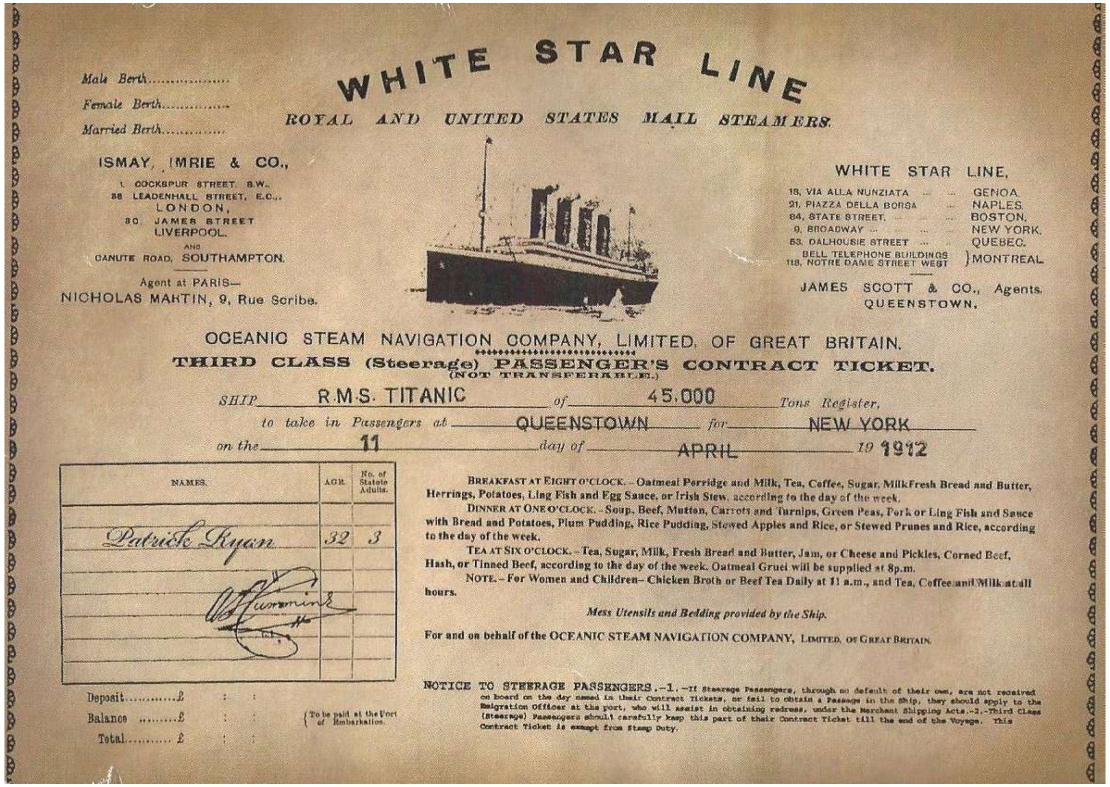
[illegible]

https://ko.wikipedia.org/wiki/RMS_타이타닉#/media/파일:Titanic_lifeboat.jpg

https://ko.wikipedia.org/wiki/RMS_타이타닉#/media/파일:Titanic-New_York_Herald_front_page.jpeg

Information

- 탑승자 및 배에 대한 다양한 정보를 얻을 수 있다.



Male Berth..................

Female Berth..................

Married Berth..................

WHITE STAR LINE

ROYAL AND UNITED STATES MAIL STEAMERS.

ISMAY, IMRIE & CO.,
1, COCKSPUR STREET, S.W.,
38 LEADENHALL STREET, E.C.,
LONDON,
30, JAMES STREET
LIVERPOOL.
AND
CANUTE ROAD, SOUTHAMPTON.

Agent at PARIS—
NICHOLAS MARTIN, 9, Rue Scribe.

WHITE STAR LINE.

18, VIA ALLA NUNZIATA	GENOA.
21, PIAZZA DELLA BORSA ...	NAPLES.
84, STATE STREET.	BOSTON.
9, BROADWAY	NEW YORK.
53, DALHOUSIE STREET	QUEBEC.
BELL TELEPHONE BUILDINGS 118, NOTRE DAME STREET WEST	MONTREAL.

JAMES SCOTT & CO., Agents.
QUEENSTOWN.

OCEANIC STEAM NAVIGATION COMPANY, LIMITED, OF GREAT BRITAIN.

THIRD CLASS (Steerage) PASSENGER'S CONTRACT TICKET.
(NOT TRANSFERABLE.)

SHIP R.M.S. TITANIC *of* 45,000 *Tons Register,*
to take in Passengers at QUEENSTOWN *for* NEW YORK
on the 11 *day of* APRIL *19* 1912

NAMES.	AGE.	No. of Statute Adults.
Patrick Ryan	32	3

Deposit............£ : :
Balance ....£ : : { To be paid at the Port of Embarkation.
Total............ £ : :

BREAKFAST AT EIGHT O'CLOCK.—Oatmeal Porridge and Milk, Tea, Coffee, Sugar, MilkFresh Bread and Butter, Herrings, Potatoes, Ling Fish and Egg Sauce, or Irish Stew, according to the day of the week.

DINNER AT ONE O'CLOCK.—Soup, Beef, Mutton, Carrots and Turnips, Green Peas, Pork or Ling Fish and Sauce with Bread and Potatoes, Plum Pudding, Rice Pudding, Stewed Apples and Rice, or Stewed Prunes and Rice, according to the day of the week.

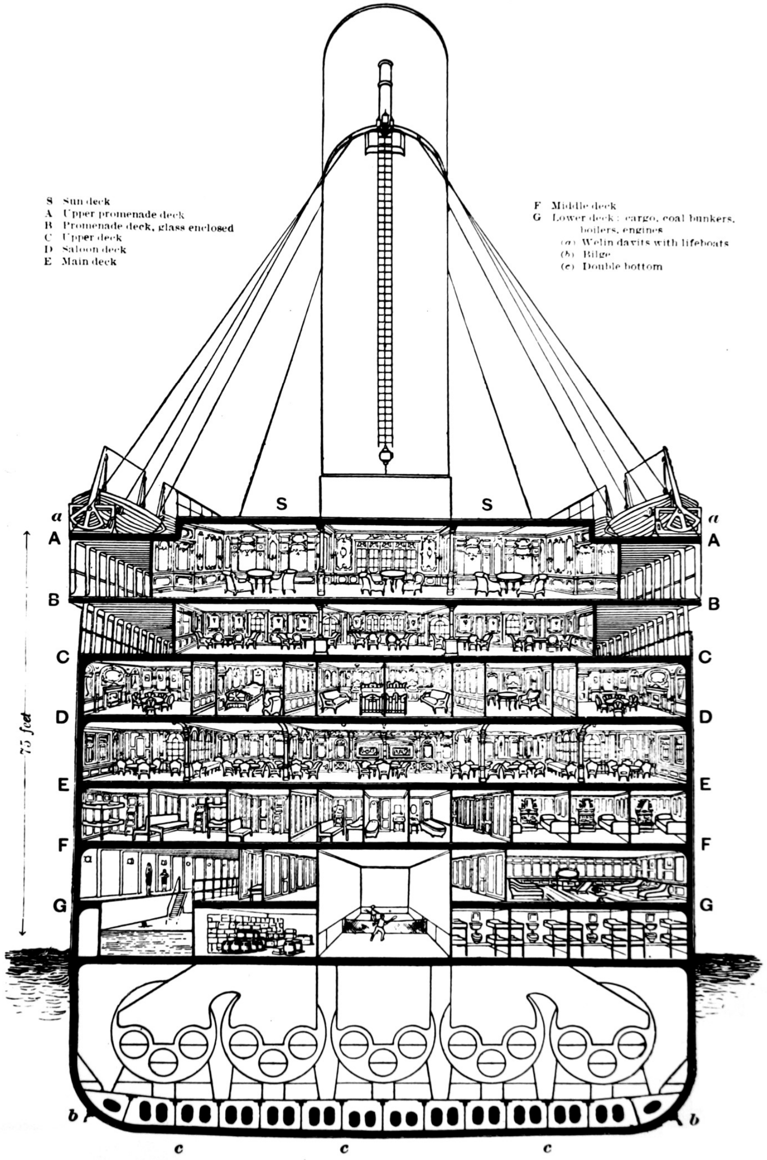
TEA AT SIX O'CLOCK.—Tea, Sugar, Milk, Fresh Bread and Butter, Jam, or Cheese and Pickles, Corned Beef, Hash, or Tinned Beef, according to the day of the week. Oatmeal Gruel will be supplied at 8p.m.

NOTE.—For Women and Children—Chicken Broth or Beef Tea Daily at 11 a.m., and Tea, Coffee and Milk at all hours.

Mess Utensils and Bedding provided by the Ship.

For and on behalf of the OCEANIC STEAM NAVIGATION COMPANY, LIMITED, OF GREAT BRITAIN.

NOTICE TO STEERAGE PASSENGERS.—1.—If Steerage Passengers, through no default of their own, are not received on board on the day named in their Contract Tickets, or fail to obtain a Passage in the Ship, they should apply to the Emigration Officer at the port, who will assist in obtaining redress, under the Merchant Shipping Acts.—2.—Third Class (Steerage) Passengers should carefully keep this part of their Contract Ticket till the end of the Voyage. This Contract Ticket is exempt from Stamp Duty.



S Sun deck
A Upper promenade deck
B Promenade deck, glass enclosed
C Upper deck
D Saloon deck
E Main deck

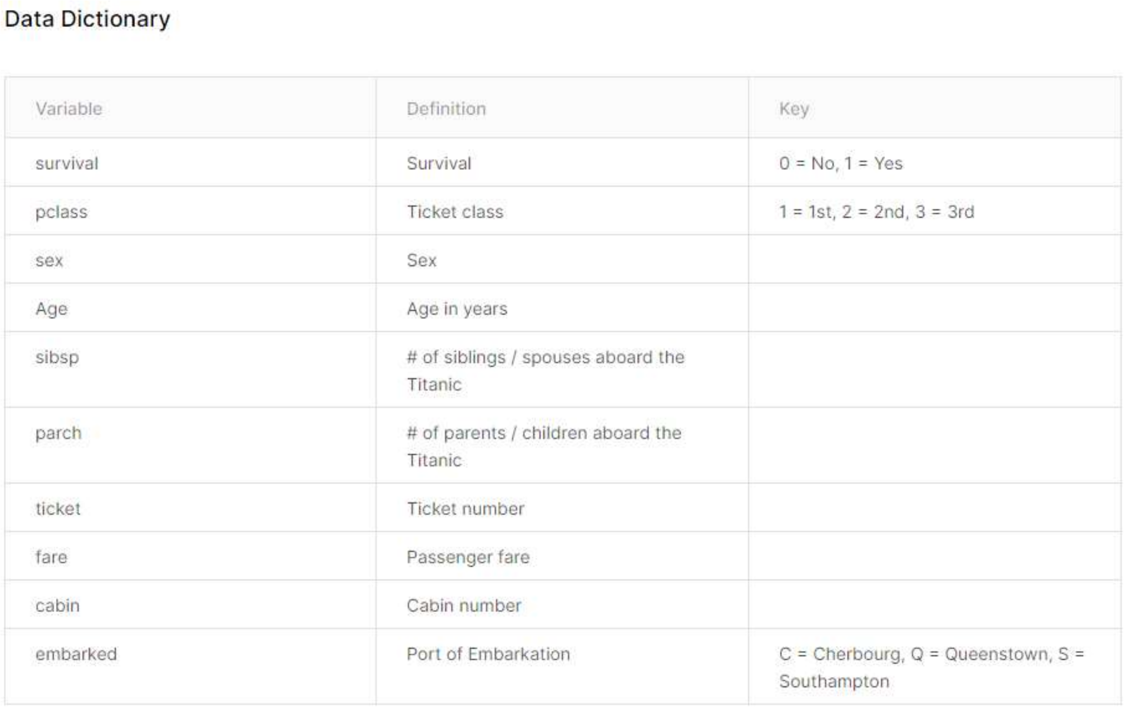
F Middle deck
G Lower deck: cargo, coal bunkers, boilers, engines
(a) Welin davits with lifeboats
(b) Bilge
(c) Double bottom

https://twitter.com/weird_hist/status/912449385346281472

https://upload.wikimedia.org/wikipedia/commons/8/84/Titanic_cutaway_diagram.png

Problem

- Predict survival on the Titanic and get familiar with ML basics



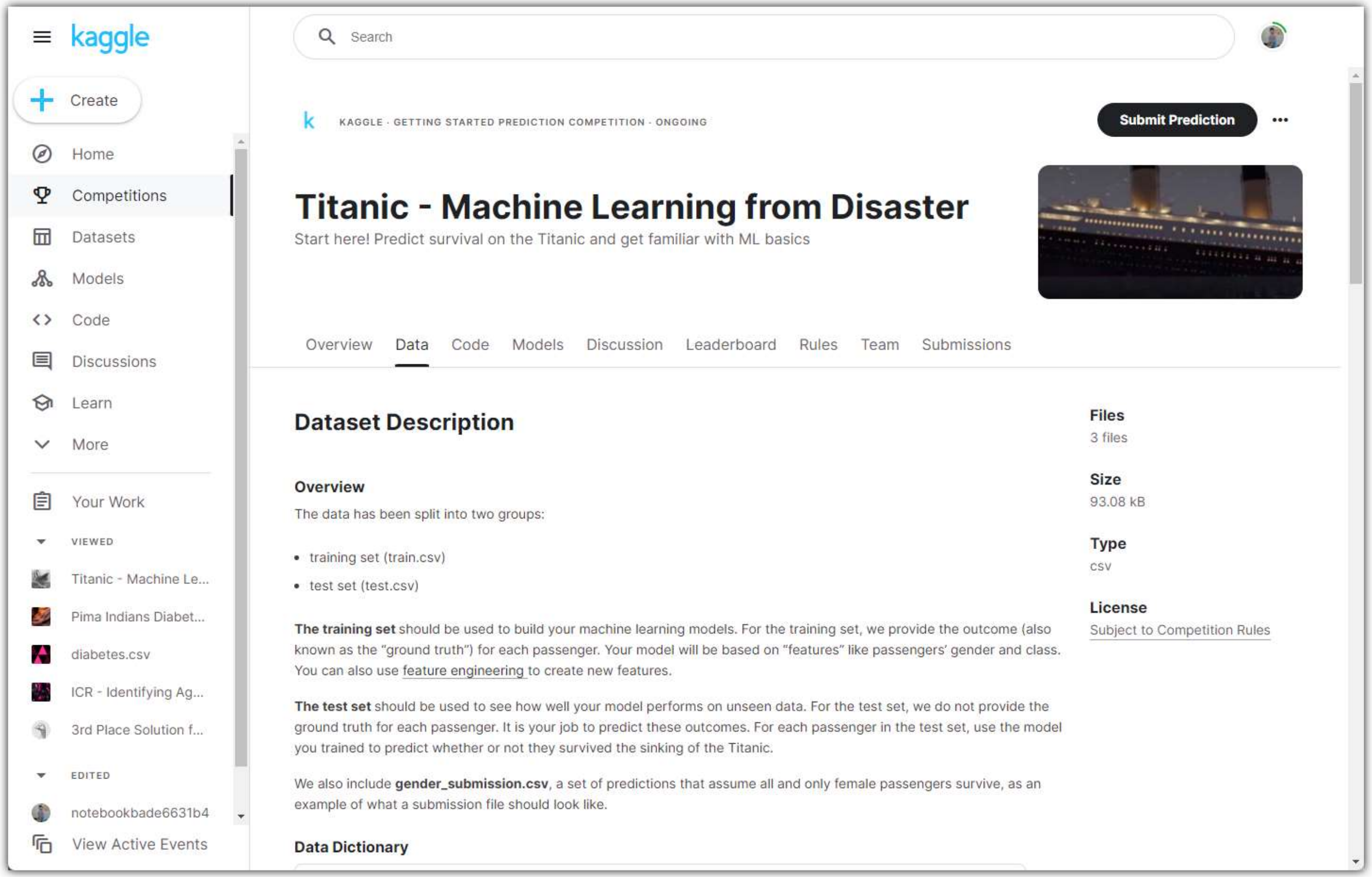
Data Dictionary

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

https://www.kaggle.com/c/titanic/data

https://en.wikipedia.org/wiki/Rules_of_Survival#/media/File:Rules_of_Survival_Google_Play_Logo.png

□ Kaggle



Titanic - Machine Learning from Disaster

Start here! Predict survival on the Titanic and get familiar with ML basics

Dataset Description

Overview

The data has been split into two groups:

- training set (train.csv)
- test set (test.csv)

The training set should be used to build your machine learning models. For the training set, we provide the outcome (also known as the "ground truth") for each passenger. Your model will be based on "features" like passengers' gender and class. You can also use feature engineering to create new features.

The test set should be used to see how well your model performs on unseen data. For the test set, we do not provide the ground truth for each passenger. It is your job to predict these outcomes. For each passenger in the test set, use the model you trained to predict whether or not they survived the sinking of the Titanic.

We also include **gender_submission.csv**, a set of predictions that assume all and only female passengers survive, as an example of what a submission file should look like.

Data Dictionary

Files
3 files

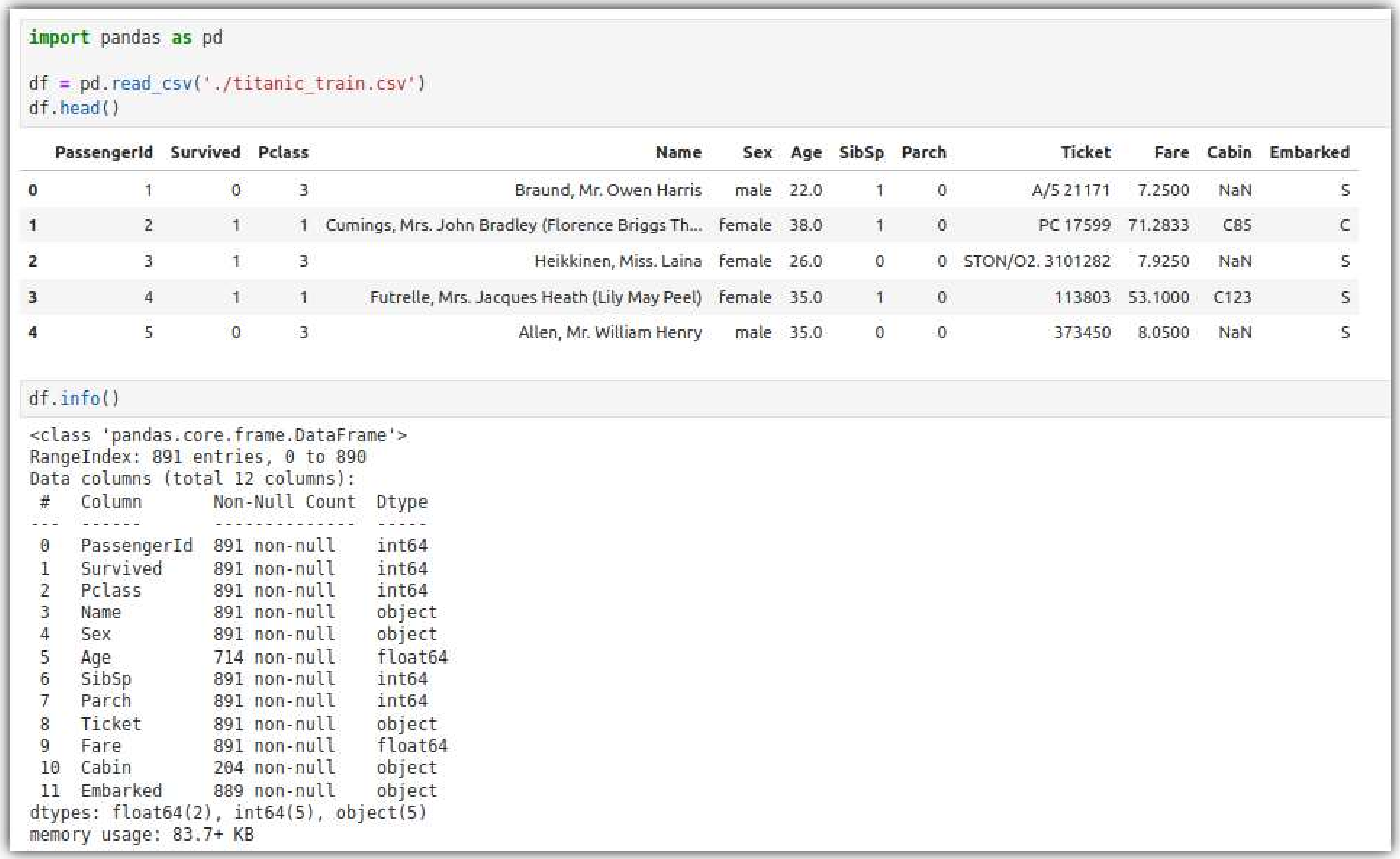
Size
93.08 kB

Type
csv

License
Subject to Competition Rules

※ 출처 : https://www.kaggle.com/competitions/titanic/data

□ 기본 정보 확인



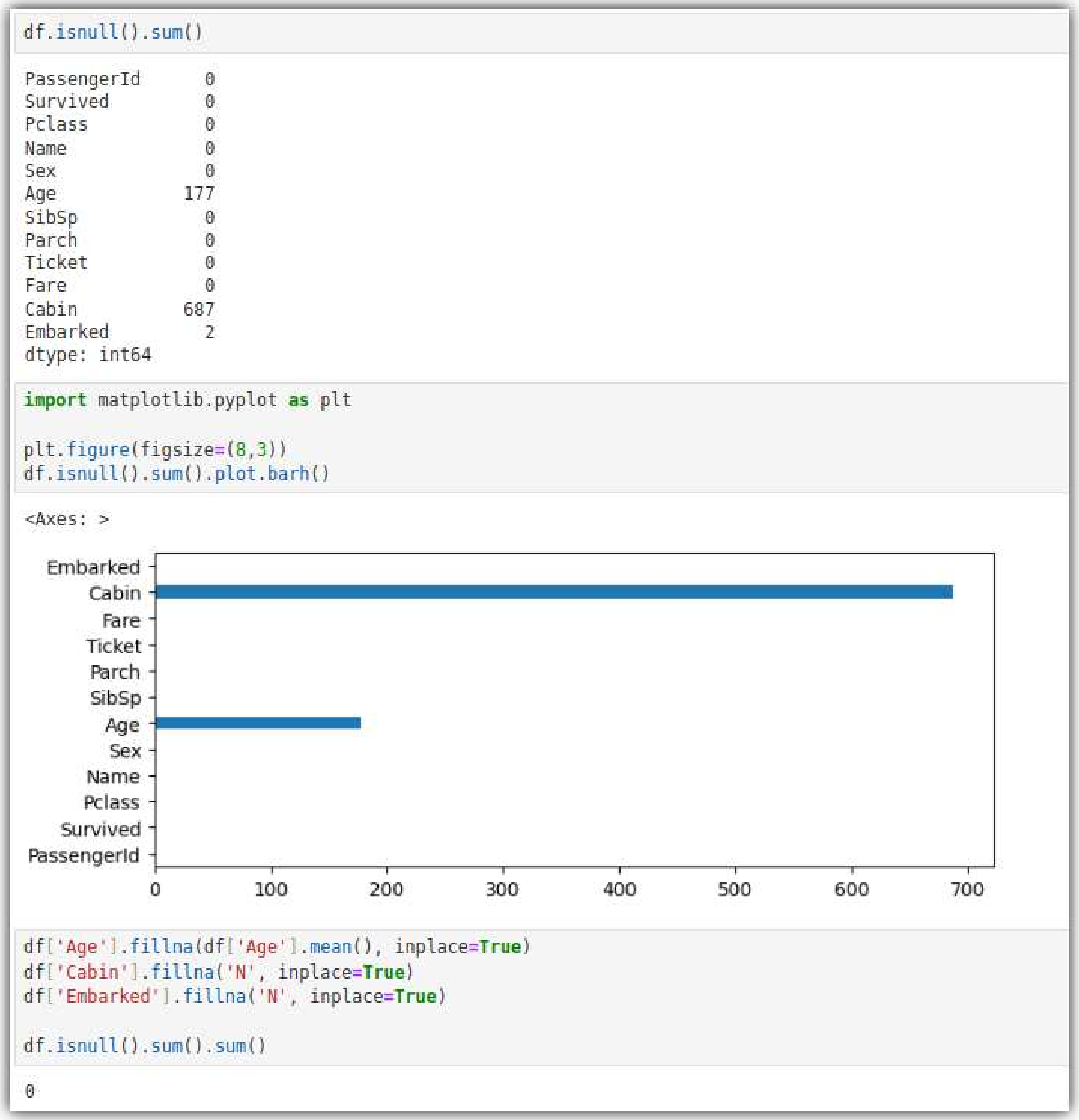
```
import pandas as pd

df = pd.read_csv('./titanic_train.csv')
df.head()
```

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

□ 결측치 처리



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

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

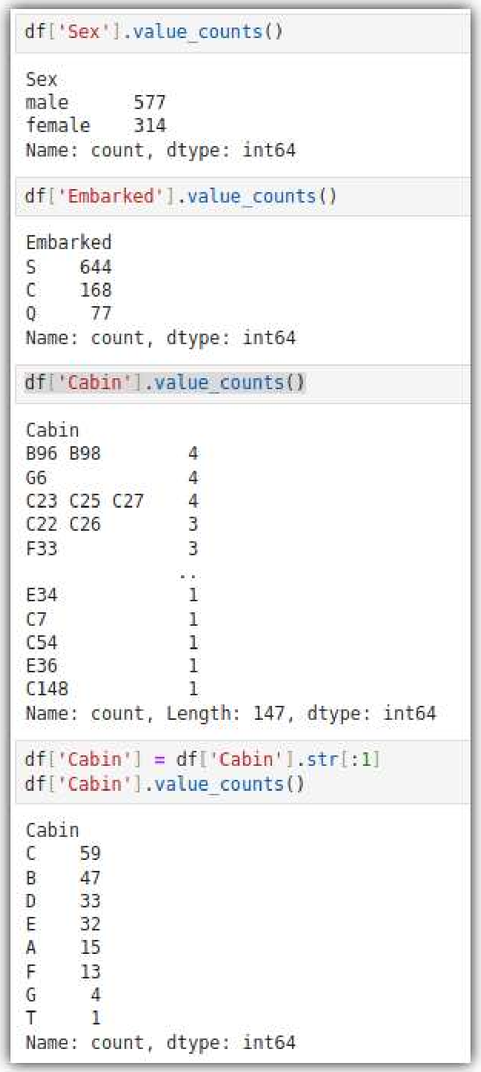
```
import matplotlib.pyplot as plt

plt.figure(figsize=(8,3))
df.isnull().sum().plot.barh()
```

```
<Axes: >
```

```
df['Age'].fillna(df['Age'].mean(), inplace=True)
df['Cabin'].fillna('N', inplace=True)
df['Embarked'].fillna('N', inplace=True)

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

```
0
```

□ 문자열 처리



```python
df['Sex'].value_counts()
```

```
Sex
male      577
female    314
Name: count, dtype: int64
```

```python
df['Embarked'].value_counts()
```

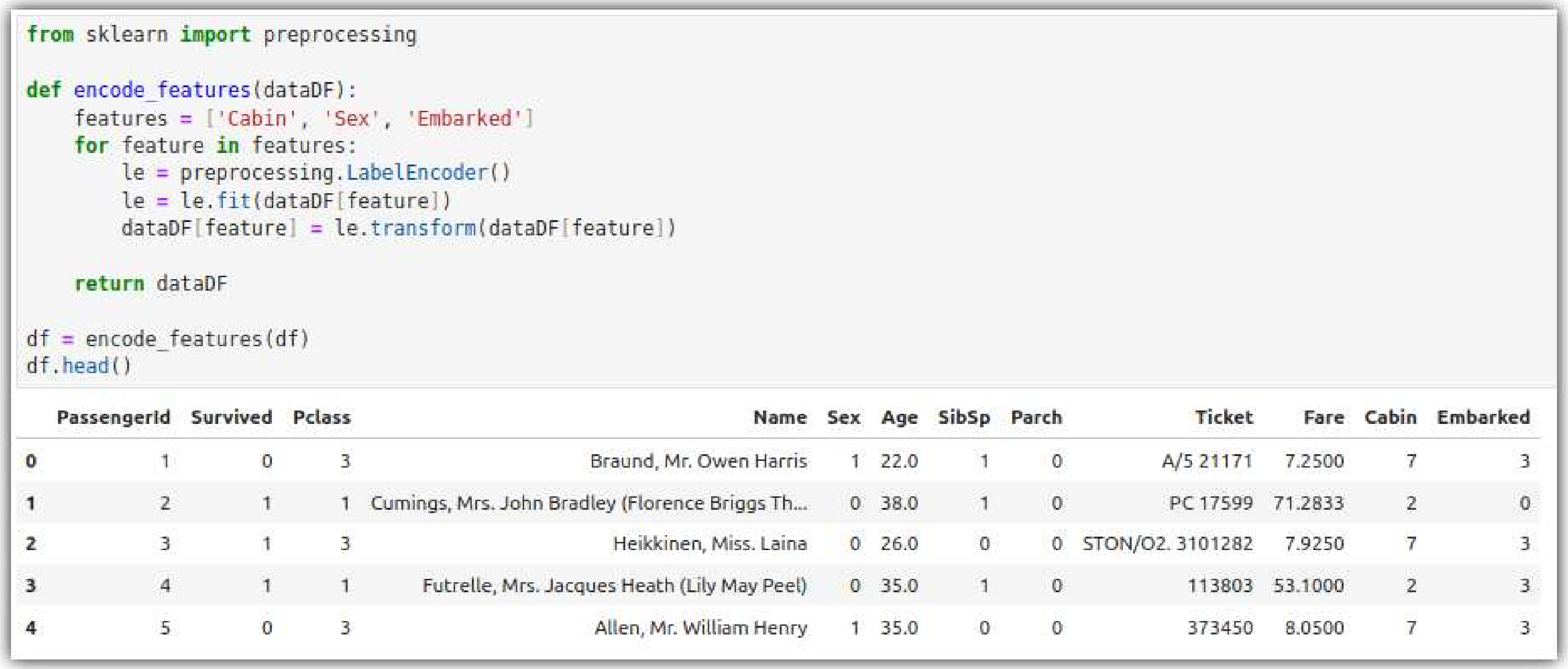
```
Embarked
S    644
C    168
Q     77
Name: count, dtype: int64
```

```python
df['Cabin'].value_counts()
```

```
Cabin
B96 B98        4
G6             4
C23 C25 C27    4
C22 C26        3
F33            3
              ..
E34            1
C7             1
C54            1
E36            1
C148           1
Name: count, Length: 147, dtype: int64
```

```python
df['Cabin'] = df['Cabin'].str[:1]
df['Cabin'].value_counts()
```

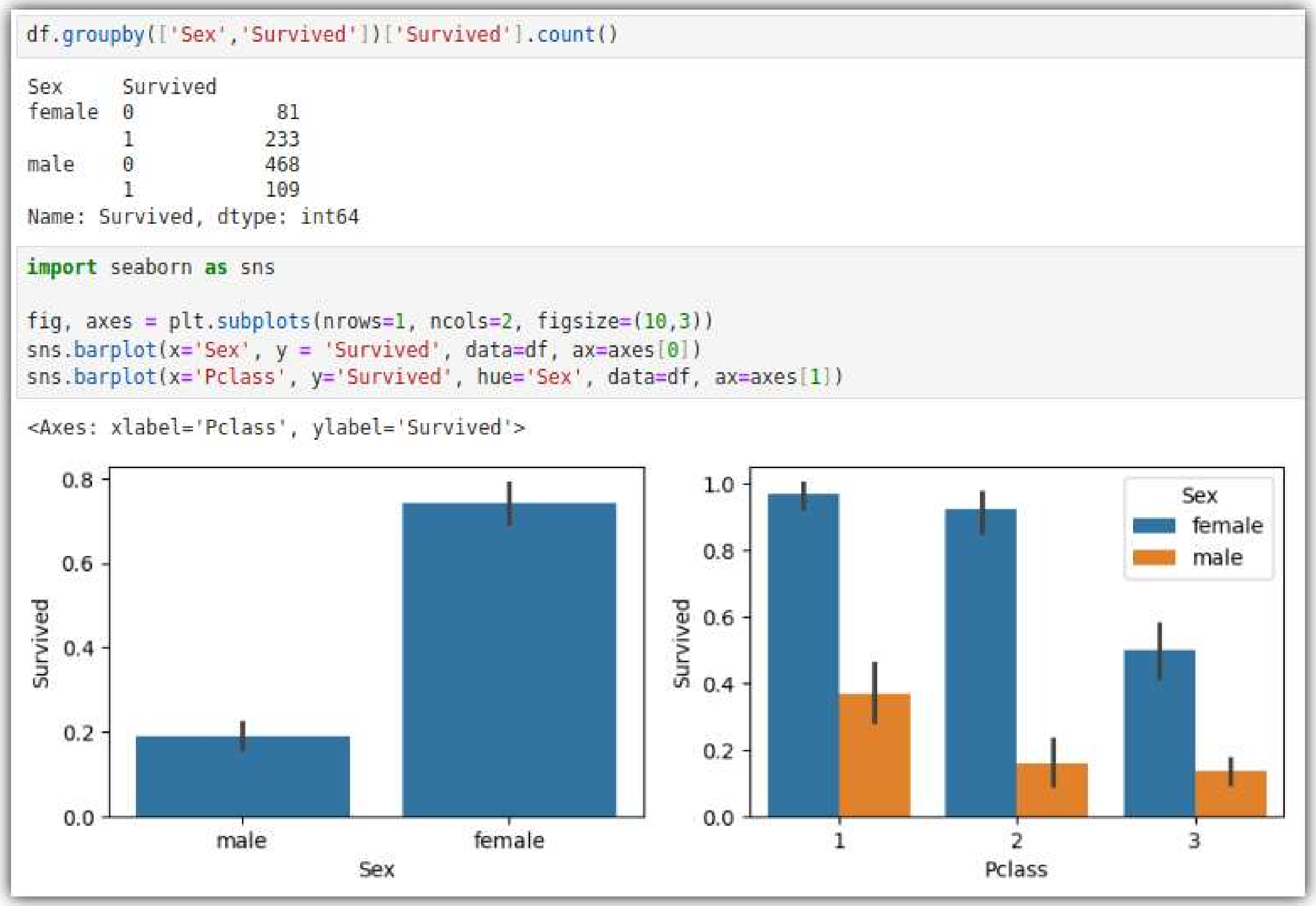
```
Cabin
C    59
B    47
D    33
E    32
A    15
F    13
G     4
T     1
Name: count, dtype: int64
```



```python
from sklearn import preprocessing

def encode_features(dataDF):
    features = ['Cabin', 'Sex', 'Embarked']
    for feature in features:
        le = preprocessing.LabelEncoder()
        le = le.fit(dataDF[feature])
        dataDF[feature] = le.transform(dataDF[feature])

    return dataDF

df = encode_features(df)
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500	7	3
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	71.2833	2	0
2	3	1	3	Heikkinen, Miss. Laina	0	26.0	0	0	STON/O2. 3101282	7.9250	7	3
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35.0	1	0	113803	53.1000	2	3
4	5	0	3	Allen, Mr. William Henry	1	35.0	0	0	373450	8.0500	7	3

□ 데이터 탐색



```
df.groupby(['Sex','Survived'])['Survived'].count()
```

```
Sex     Survived
female  0            81
        1           233
male    0           468
        1           109
Name: Survived, dtype: int64
```

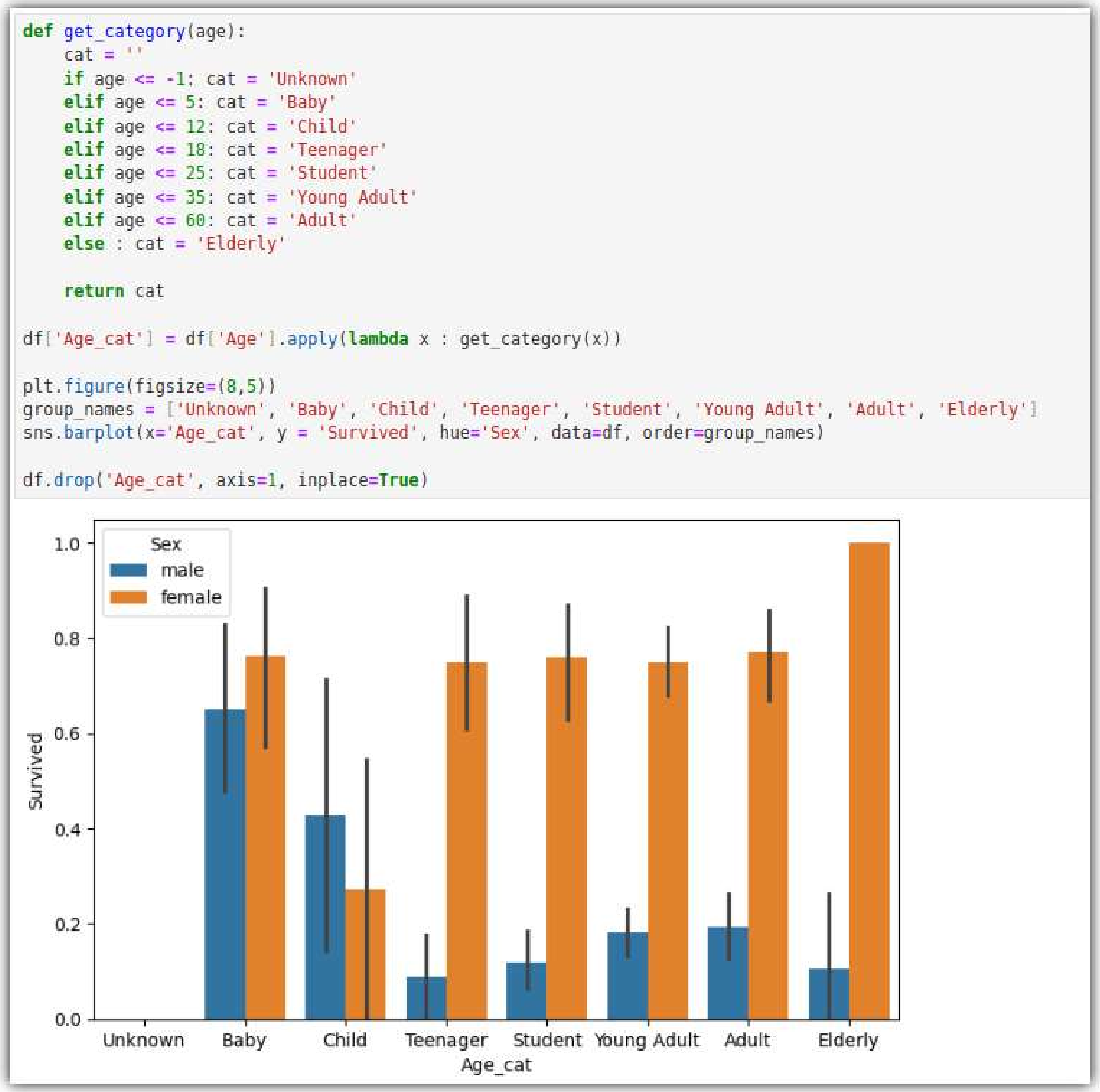
```
import seaborn as sns

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10,3))
sns.barplot(x='Sex', y = 'Survived', data=df, ax=axes[0])
sns.barplot(x='Pclass', y='Survived', hue='Sex', data=df, ax=axes[1])
```

```
<Axes: xlabel='Pclass', ylabel='Survived'>
```

선실 등급에 따른 생존율

여성의 생존율

□ 카테고리化

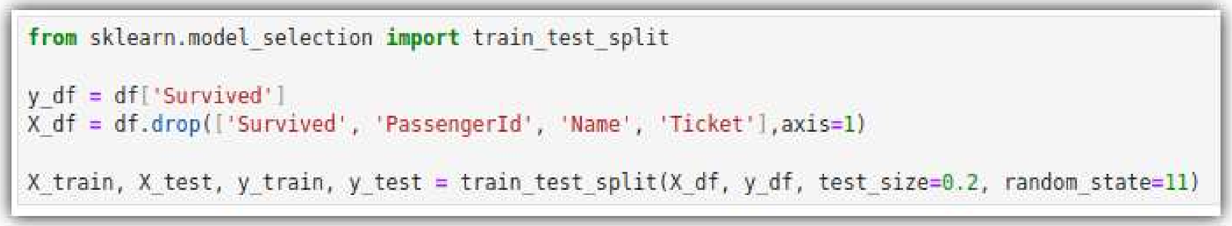


```
def get_category(age):
    cat = ''
    if age <= -1: cat = 'Unknown'
    elif age <= 5: cat = 'Baby'
    elif age <= 12: cat = 'Child'
    elif age <= 18: cat = 'Teenager'
    elif age <= 25: cat = 'Student'
    elif age <= 35: cat = 'Young Adult'
    elif age <= 60: cat = 'Adult'
    else : cat = 'Elderly'

    return cat

df['Age_cat'] = df['Age'].apply(lambda x : get_category(x))

plt.figure(figsize=(8,5))
group_names = ['Unknown', 'Baby', 'Child', 'Teenager', 'Student', 'Young Adult', 'Adult', 'Elderly']
sns.barplot(x='Age_cat', y = 'Survived', hue='Sex', data=df, order=group_names)

df.drop('Age_cat', axis=1, inplace=True)
```

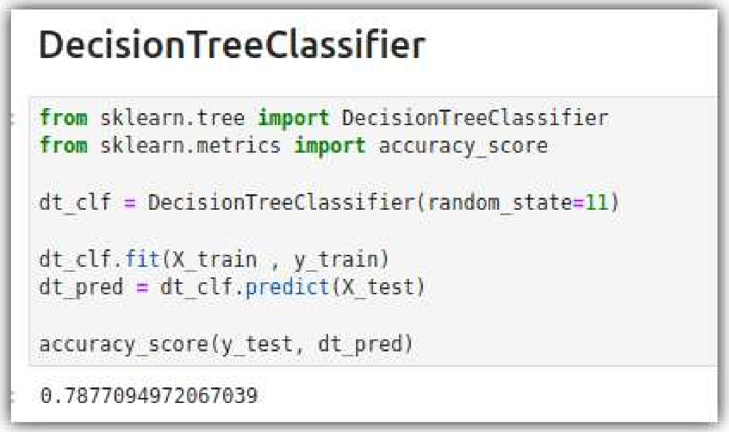
어린 아이들의 생존율

여성의 생존율

□ Training



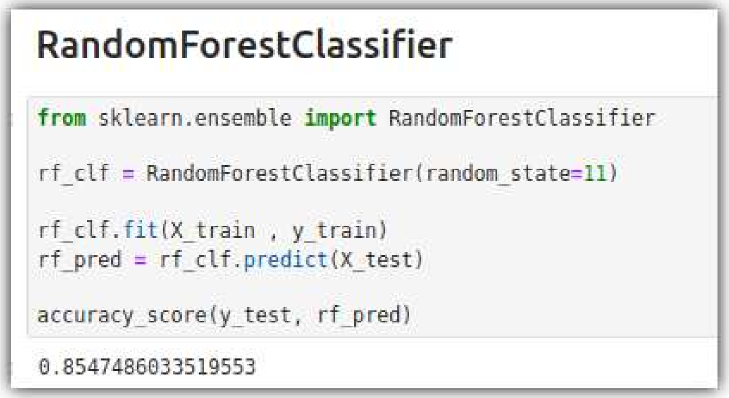
```
from sklearn.model_selection import train_test_split

y_df = df['Survived']
X_df = df.drop(['Survived', 'PassengerId', 'Name', 'Ticket'],axis=1)

X_train, X_test, y_train, y_test = train_test_split(X_df, y_df, test_size=0.2, random_state=11)
```



DecisionTreeClassifier

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

dt_clf = DecisionTreeClassifier(random_state=11)

dt_clf.fit(X_train , y_train)
dt_pred = dt_clf.predict(X_test)

accuracy_score(y_test, dt_pred)
```

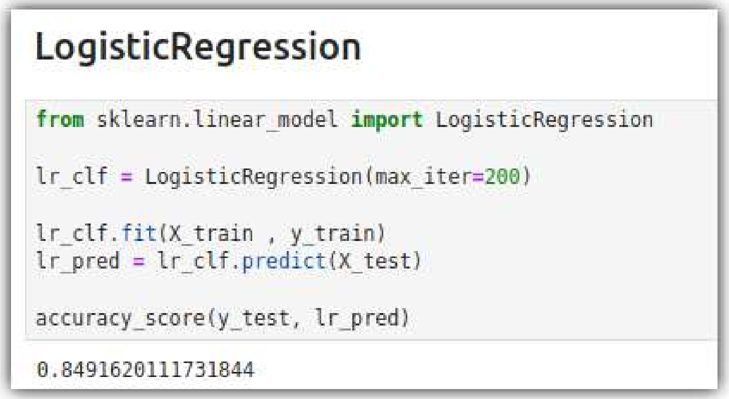
```
0.7877094972067039
```



RandomForestClassifier

```
from sklearn.ensemble import RandomForestClassifier

rf_clf = RandomForestClassifier(random_state=11)

rf_clf.fit(X_train , y_train)
rf_pred = rf_clf.predict(X_test)

accuracy_score(y_test, rf_pred)
```

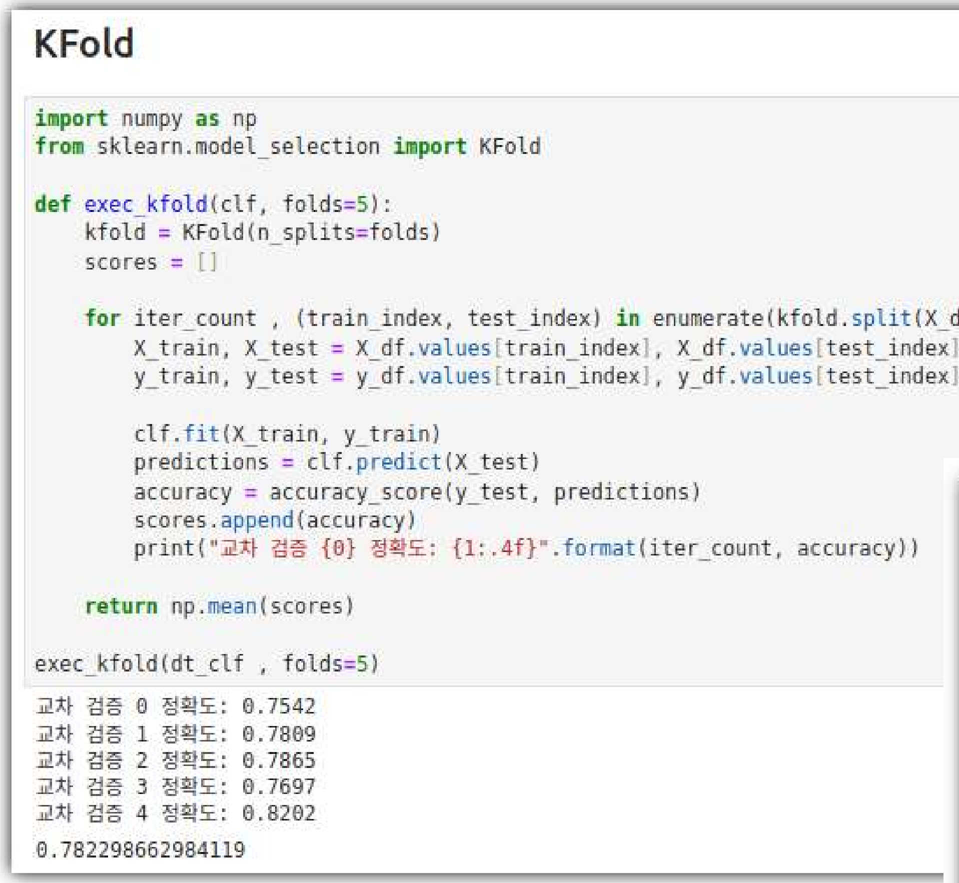
```
0.8547486033519553
```



LogisticRegression

```
from sklearn.linear_model import LogisticRegression

lr_clf = LogisticRegression(max_iter=200)

lr_clf.fit(X_train , y_train)
lr_pred = lr_clf.predict(X_test)

accuracy_score(y_test, lr_pred)
```

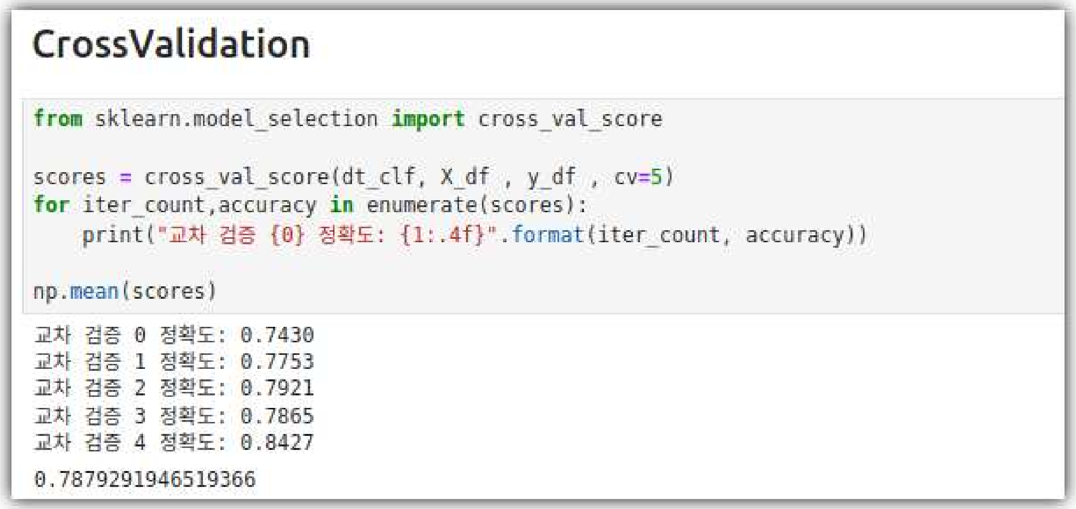
```
0.8491620111731844
```



KFold

```
import numpy as np
from sklearn.model_selection import KFold

def exec_kfold(clf, folds=5):
    kfold = KFold(n_splits=folds)
    scores = []

    for iter_count , (train_index, test_index) in enumerate(kfold.split(X_df)):
        X_train, X_test = X_df.values[train_index], X_df.values[test_index]
        y_train, y_test = y_df.values[train_index], y_df.values[test_index]

        clf.fit(X_train, y_train)
        predictions = clf.predict(X_test)
        accuracy = accuracy_score(y_test, predictions)
        scores.append(accuracy)
        print("교차 검증 {0} 정확도: {1:.4f}".format(iter_count, accuracy))

    return np.mean(scores)

exec_kfold(dt_clf , folds=5)
```

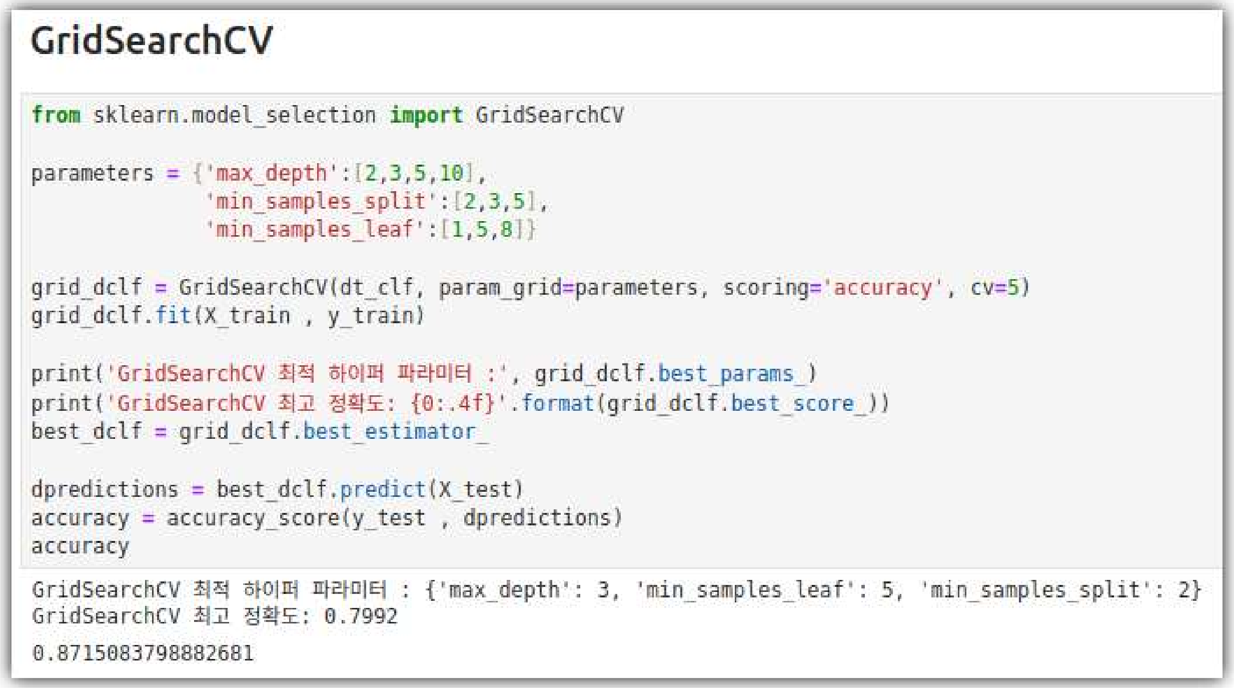
```
교차 검증 0 정확도: 0.7542
교차 검증 1 정확도: 0.7809
교차 검증 2 정확도: 0.7865
교차 검증 3 정확도: 0.7697
교차 검증 4 정확도: 0.8202
0.782298662984119
```



CrossValidation

```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(dt_clf, X_df , y_df , cv=5)
for iter_count,accuracy in enumerate(scores):
    print("교차 검증 {0} 정확도: {1:.4f}".format(iter_count, accuracy))

np.mean(scores)
```

```
교차 검증 0 정확도: 0.7430
교차 검증 1 정확도: 0.7753
교차 검증 2 정확도: 0.7921
교차 검증 3 정확도: 0.7865
교차 검증 4 정확도: 0.8427
0.7879291946519366
```



GridSearchCV

```
from sklearn.model_selection import GridSearchCV

parameters = {'max_depth':[2,3,5,10],
              'min_samples_split':[2,3,5],
              'min_samples_leaf':[1,5,8]}

grid_dclf = GridSearchCV(dt_clf, param_grid=parameters, scoring='accuracy', cv=5)
grid_dclf.fit(X_train , y_train)

print('GridSearchCV 최적 하이퍼 파라미터 :', grid_dclf.best_params_)
print('GridSearchCV 최고 정확도: {0:.4f}'.format(grid_dclf.best_score_))
best_dclf = grid_dclf.best_estimator_

dpredictions = best_dclf.predict(X_test)
accuracy = accuracy_score(y_test , dpredictions)
accuracy
```

```
GridSearchCV 최적 하이퍼 파라미터 : {'max_depth': 3, 'min_samples_leaf': 5, 'min_samples_split': 2}
GridSearchCV 최고 정확도: 0.7992
0.8715083798882681
```

You’ve really worked hard today

Next Week ~ ?