Machine Learning with Python

Life is too short, You need Python

실습 내용

- 머신러닝 모델링을 위한 코딩은 무조건 할 수 있어야 합니다.
- 코딩 내용을 자세히 알지 못해도 무작정 코딩을 진행해봅니다.
- Happy 데이터를 대상으로 모델링을 진행합니다.
- LinearRegression 알고리즘을 사용합니다.
- 다양한 방법으로 모델 성능을 평가합니다.

1.환경 준비

- 기본 **라이브러리**와 대상 **데이터**를 가져와 이후 과정을 준비합니다.

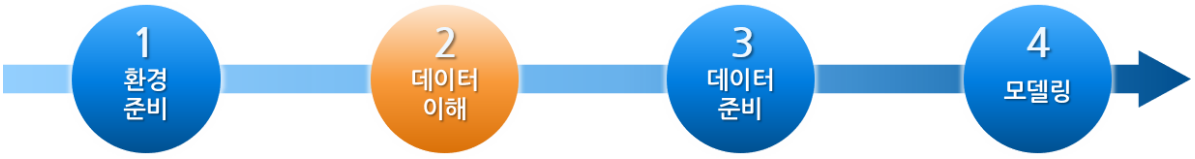
In [1]:
```
# 라이브러리 불러오기
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
```

```
warnings.filterwarnings(action='ignore')
%config InlineBackend.figure_format = 'retina'
```

In [2]:
```
# 데이터 읽어오기
path = 'https://raw.githubusercontent.com/Jangrae/csv/master/income_happy.csv'
data = pd.read_csv(path)
```

2.데이터 이해

- 분석할 데이터를 **충분히 이해**할 수 있도록 다양한 **탐색** 과정을 수행합니다.



In [3]:
```
# 상위 몇 개 행 확인
data.head()
```

Out[3]:

	income	happiness
0	3.862647	2.314489
1	4.979381	3.433490
2	4.923957	4.599373
3	3.214372	2.791114
4	7.196409	5.596398

데이터 설명

- income: 수입 (단위: 10,000$)
- happiness: 행복 정도 (1~ 10)

In [4]:
```
# 하위 몇 개 행 확인
data.tail()
```

Out[4]:

	income	happiness
493	5.249209	4.568705
494	3.471799	2.535002
495	6.087610	4.397451
496	3.440847	2.070664
497	4.530545	3.710193

In [5]:
```
# 변수 확인
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 498 entries, 0 to 497
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   income     498 non-null    float64
 1   happiness  498 non-null    float64
dtypes: float64(2)
memory usage: 7.9 KB
```

In [6]:
```
# 데이터 크기 확인
data.shape
```

Out[6]:
```
(498, 2)
```

In [7]:
```
# 기술통계 확인
data.describe().T
```

Out[7]:

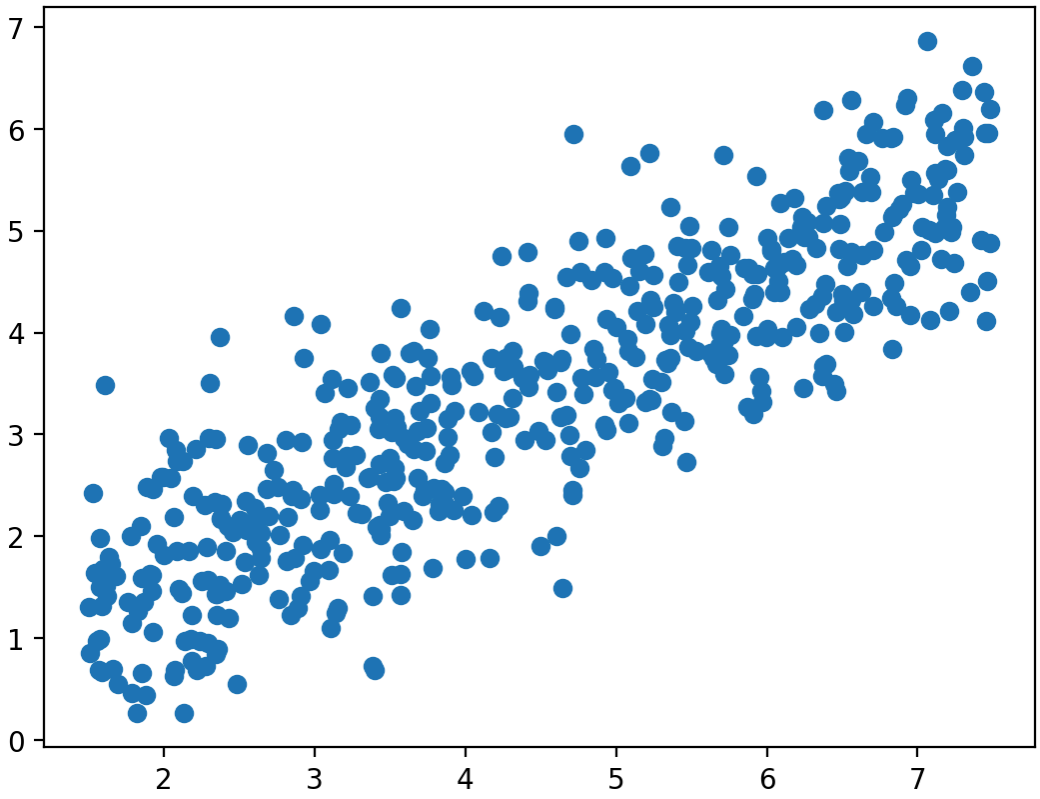
	count	mean	std	min	25%	50%	75%	max
income	498.0	4.466902	1.737527	1.506275	3.006256	4.423710	5.991913	7.481521
happiness	498.0	3.392859	1.432813	0.266044	2.265864	3.472536	4.502621	6.863388

In [8]:
```
# 상관관계 확인
data.corr()
```

Out[8]:

	income	happiness
income	1.000000	0.865634
happiness	0.865634	1.000000

In [23]:
```
# 산점도
plt.scatter(x='income', y='happiness', data=data)
plt.show()
```



3.데이터 준비

- **전처리** 과정을 통해 머신러닝 알고리즘에 사용할 수 있는 형태의 데이터를 준비합니다.

1) x, y 분리

- 우선 target 변수를 명확히 지정합니다.
- target을 제외한 나머지 변수들 데이터는 x로 선언합니다.
- target 변수 데이터는 y로 선언합니다.
- 이 결과로 만들어진 x는 데이터프레임, y는 시리즈가 됩니다.
- 이후 모든 작업은 x, y를 대상으로 진행합니다.

In [9]:
```
# target 확인
target = 'happiness'

# 데이터 분리
x = data.drop(target, axis=1)
y = data.loc[:, target]
```

2) 학습용, 평가용 데이터 분리

- 학습용, 평가용 데이터를 적절한 비율로 분리합니다.

- 반복 실행 시 동일한 결과를 얻기 위해 random_state 옵션을 지정합니다.

In [10]:
```
# 모듈 불러오기
from sklearn.model_selection import train_test_split

# 7:3으로 분리
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=1)
```

4.모델링

- 본격적으로 모델을 **선언**하고 **학습**하고 **평가**하는 과정을 진행합니다.
- 우선 **회귀** 문제인지 **분류** 문제인지 명확히 구분합니다.

- 회귀 문제 인가요? 분류 문제인가요?
- 회귀인지 분류인지에 따라 사용할 알고리즘과 평가 방법이 달라집니다.
- 우선 다음 알고리즘을 사용합니다.
  - 알고리즘: LinearRegression

In [11]:
```
# 1단계: 불러오기
from sklearn.linear_model import LinearRegression
```

In [12]:
```
# 2단계: 선언하기
model = LinearRegression()
```

In [13]:
```
# 3단계: 학습하기
model.fit(x_train, y_train)
```

Out[13]:
```
▾ LinearRegression
LinearRegression()
```

In [14]:
```
# 4단계: 예측하기
y_pred = model.predict(x_test)
```

5.회귀 성능 평가

- 다양한 성능 지표로 회귀 모델 성능을 평가합니다.

1) MAE(Mean Absolute Error)

In [15]:
```
# 모듈 불러오기
from sklearn.metrics import mean_absolute_error
```

```
# 성능 평가
print('MAE:', mean_absolute_error(y_test, y_pred))
```

```
MAE: 0.5981154412391132
```

2) MSE(Mean Squared Error)

In [16]:
```
# 모듈 불러오기
from sklearn.metrics import mean_squared_error

# 성능 평가
print('MSE:', mean_squared_error(y_test, y_pred))
```

```
MSE: 0.5553820457607622
```

3) RMSE(Root Mean Squared Error)

In [17]:
```
# 모듈 불러오기
from sklearn.metrics import mean_squared_error

# 성능 평가
print('RMSE:', mean_squared_error(y_test, y_pred)**0.5)
```

```
RMSE: 0.745239589501767
```

4) MAPE(Mean Absolute Percentage Error)

In [18]:
```
# 모듈 불러오기
from sklearn.metrics import mean_absolute_percentage_error

# 성능 평가
print('MAP:', mean_absolute_percentage_error(y_test, y_pred))
```

```
MAP: 0.27040543725963717
```

5) R2-Score

In [19]:
```
# 모듈 불러오기
from sklearn.metrics import r2_score

# 성능 평가
print('R2:', r2_score(y_test, y_pred))
```

```
R2: 0.7324646979280791
```

In [21]:
```
# 학습, 평가 성능 비교
print("학습성능:", model.score(x_train, y_train))
print("평가성능:", model.score(x_test, y_test))
```

```
학습성능: 0.7546807786281873
평가성능: 0.7324646979280791
```

In [ ]: