Machine Learning with Python

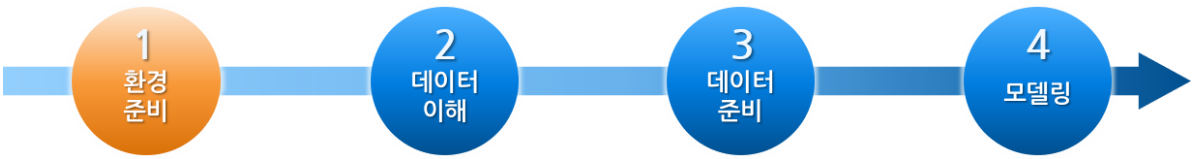
Life is too short, You need Python

실습 내용

- 머신러닝 모델링을 위한 코딩은 무조건 할 수 있어야 합니다.
- 코딩 내용을 자세히 알지 못해도 **무작정** 코딩을 진행해봅니다.
- AirQuality 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.



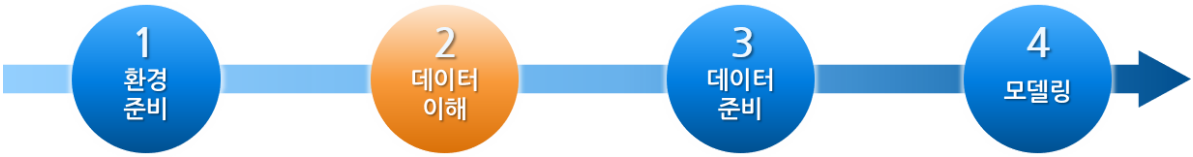
In [1]:
```
# 라이브러리 불러오기
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
```

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

In [2]:
```
# 데이터 읽어오기
path = 'https://raw.githubusercontent.com/Jangrae/csv/master/airquality_simple.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]:

	Ozone	Solar.R	Wind	Temp	Month	Day
0	41	190.0	7.4	67	5	1
1	36	118.0	8.0	72	5	2
2	12	149.0	12.6	74	5	3
3	18	313.0	11.5	62	5	4
4	19	NaN	14.3	56	5	5

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

Out[4]:

	Ozone	Solar.R	Wind	Temp	Month	Day
148	30	193.0	6.9	70	9	26
149	23	145.0	13.2	77	9	27
150	14	191.0	14.3	75	9	28
151	18	131.0	8.0	76	9	29
152	20	223.0	11.5	68	9	30

In [5]:
```
# 변수 확인
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 153 entries, 0 to 152
Data columns (total 6 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   Ozone    153 non-null    int64  
 1   Solar.R  146 non-null    float64
 2   Wind     153 non-null    float64
 3   Temp     153 non-null    int64  
 4   Month    153 non-null    int64  
 5   Day      153 non-null    int64  
dtypes: float64(2), int64(4)
memory usage: 7.3 KB
```

In [6]:
```
# 기술통계 확인
data.describe()
```

Out[6]:

	Ozone	Solar.R	Wind	Temp	Month	Day
count	153.000000	146.000000	153.000000	153.000000	153.000000	153.000000
mean	42.052288	185.931507	9.957516	77.882353	6.993464	15.803922
std	30.156127	90.058422	3.523001	9.465270	1.416522	8.864520
min	1.000000	7.000000	1.700000	56.000000	5.000000	1.000000
25%	20.000000	115.750000	7.400000	72.000000	6.000000	8.000000
50%	34.000000	205.000000	9.700000	79.000000	7.000000	16.000000
75%	59.000000	258.750000	11.500000	85.000000	8.000000	23.000000
max	168.000000	334.000000	20.700000	97.000000	9.000000	31.000000

In [7]:
```
# 상관관계 확인
data.corr(numeric_only=True)
```

Out[7]:

	Ozone	Solar.R	Wind	Temp	Month	Day
Ozone	1.000000	0.280068	-0.605478	0.683372	0.174197	0.004419
Solar.R	0.280068	1.000000	-0.056792	0.275840	-0.075301	-0.150275
Wind	-0.605478	-0.056792	1.000000	-0.457988	-0.178293	0.027181
Temp	0.683372	0.275840	-0.457988	1.000000	0.420947	-0.130593
Month	0.174197	-0.075301	-0.178293	0.420947	1.000000	-0.007962
Day	0.004419	-0.150275	0.027181	-0.130593	-0.007962	1.000000

3.데이터 준비

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

1) 결측치 처리

- 결측치가 있으면 제거하거나 적절한 값으로 채웁니다.

In [8]:
```
# 결측치 확인
data.isnull().sum()
```

Out[8]:
```
Ozone      0
Solar.R    7
Wind       0
Temp       0
Month      0
Day        0
dtype: int64
```

In [9]:
```
# 전날 값으로 결측치 채우기
data.fillna(method='ffill', inplace=True)

# 확인
data.isnull().sum()
```

Out[9]:
```
Ozone      0
Solar.R    0
Wind       0
Temp       0
Month      0
Day        0
dtype: int64
```

2) 변수 제거

- 분석에 의미가 없다고 판단되는 변수는 제거합니다.

In [10]:
```
# 변수 제거
drop_cols = ['Month', 'Day']
data.drop(drop_cols, axis=1, inplace=True)

# 확인
data.head()
```

Out[10]:

	Ozone	Solar.R	Wind	Temp
0	41	190.0	7.4	67
1	36	118.0	8.0	72
2	12	149.0	12.6	74
3	18	313.0	11.5	62
4	19	313.0	14.3	56

3) 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 [11]:
```
# target 확인
target = 'Ozone'

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

4) 학습용, 평가용 데이터 분리

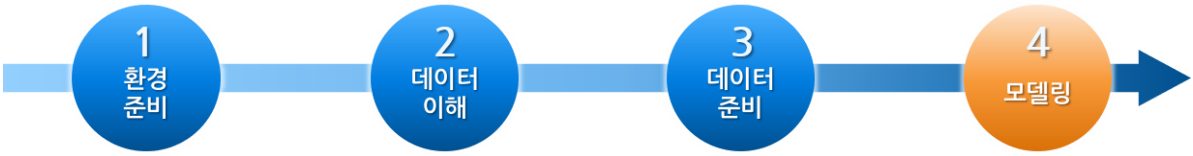
- 학습용, 평가용 데이터를 적절한 비율로 분리합니다.
- 반복 실행 시 동일한 결과를 얻기 위해 random_state 옵션을 지정합니다.

In [12]:
```
# 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, test_size=0.3, 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 [13]:
```
# 1단계: 불러오기
from sklearn.linear_model import LinearRegression
```

In [14]:
```
# 2단계: 선언하기
model = LinearRegression()
```

In [15]:
```
# 3단계: 학습하기
model.fit(x_train, y_train)
```

Out[15]:
```
▾ LinearRegression
LinearRegression()
```

In [16]:
```
# 4단계: 예측하기
y_pred = model.predict(x_test)
```

5.회귀 성능 평가

- 다양한 성능 지표로 회귀 모델 성능을 평가합니다.

1) MAE(Mean Absolute Error)

$$\large MAE=\frac{1}{n}\sum_{i=1}^{n}|y_{i}-\hat{y}_{i}|$$

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

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

```
13.976843190385708
```

2) MSE(Mean Squared Error)

$$\large MSE=\frac{1}{n}\sum_{i=1}^{n}(y_{i}-\hat{y}_{i})^2$$

In [20]:
```
# 모듈 불러오기
from sklearn.metrics import mean_squared_error

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

```
MSE: 341.67887406681893
```

3) RMSE(Root Mean Squared Error)

$$\large RMSE=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{i}-\hat{y}_{i})^2}$$

In [22]:
```
# 모듈 불러오기
from sklearn.metrics import mean_squared_error

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

```
RMSE: 18.484557718993955
```

- 0.5 제곱 대신에 squared=False 옵션을 지정해도 됩니다.

In [23]:
```
# 모듈 불러오기
from sklearn.metrics import mean_squared_error

# 성능 평가
print('RMSE:', mean_squared_error(y_test, y_pred, squared=False))
```

```
RMSE: 18.484557718993955
```

4) MAPE(Mean Absolute Percentage Error)

$$\large MAPE=\frac{1}{n}\sum_{i=1}^{n}\left |\frac{y_{i}-\hat{y}_{i}}{y_{i}}\right |$$

In [24]:
```
# 모듈 불러오기
from sklearn.metrics import mean_absolute_percentage_error

# 성능 평가
print('MAPE:', mean_absolute_percentage_error(y_test, y_pred)) # 오차의 비율
```

```
MAPE: 0.4718597698848258
```

5) R2-Score

$$\large R^2=1-\frac{SSE}{SST}=1-\frac{\sum_{i=1}^{n}(y_{i}-\hat{y}_{i})^2}{\sum_{i=1}^{n}(y_{i}-\bar{y}_{i})^2}$$

In [25]:
```
# 모듈 불러오기
from sklearn.metrics import r2_score

# 성능 평가
print('R2:', r2_score(y_test, y_pred)) # 57% 설명을 더 잘 하고 있다
```

```
R2: 0.5744131358040061
```

- score() 메서드를 사용해 R2 Score를 확인할 수 있습니다.

In [26]:
```
# 참고
model.score(x_test, y_test)
```

Out[26]:
```
0.5744131358040061
```

6) 학습 성능 확인

In [27]:
```
# 학습 데이터에 대한 예측
y_train_pred = model.predict(x_train)

# 학습 성능 확인
print('R2:', r2_score(y_train, y_train_pred)) #학습 성능과 평가 성능이 비슷 과적합은 아님
```

```
R2: 0.5764536433276649
```

In [28]:
```
print('학습성능:', model.score(x_train, y_train))
print('평가성능:', model.score(x_test, y_test))
```

```
학습성능: 0.5764536433276649
평가성능: 0.5744131358040061
```

In [ ]:
```

```