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 모델링 해서 오존 농도를 예측해 봅니다.
- LinearRegression 알고리즘을 사용합니다.

1.환경 준비

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

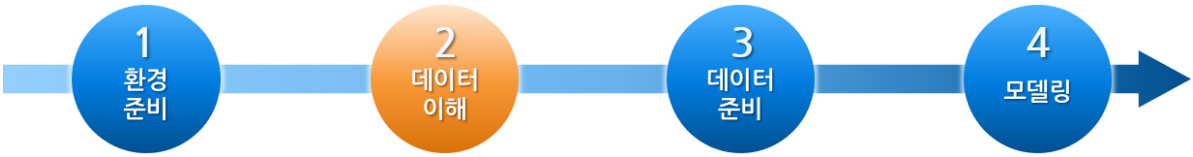
In [1]:
```
# 라이브러리 불러오기
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

warnings.filterwarnings(action='ignore') # 경고 무시
%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 [7]:
```
# 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 [8]:
```
# 기술통계 확인
data.describe().T
```

Out[8]:

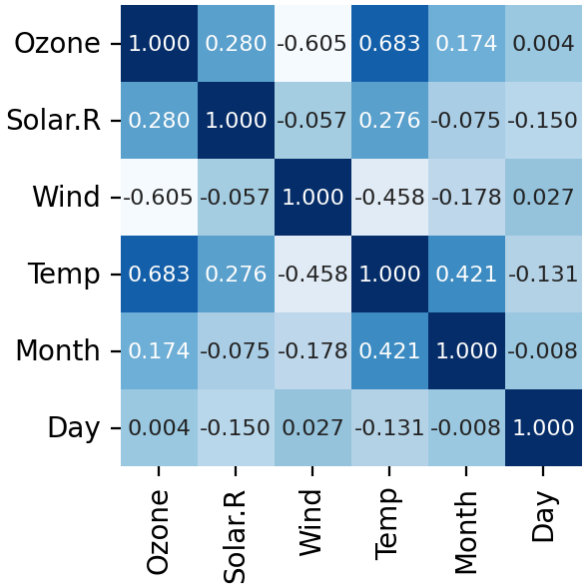
	count	mean	std	min	25%	50%	75%	max
Ozone	153.0	42.052288	30.156127	1.0	20.00	34.0	59.00	168.0
Solar.R	146.0	185.931507	90.058422	7.0	115.75	205.0	258.75	334.0
Wind	153.0	9.957516	3.523001	1.7	7.40	9.7	11.50	20.7
Temp	153.0	77.882353	9.465270	56.0	72.00	79.0	85.00	97.0
Month	153.0	6.993464	1.416522	5.0	6.00	7.0	8.00	9.0
Day	153.0	15.803922	8.864520	1.0	8.00	16.0	23.00	31.0

In [10]:
```
# 상관관계 확인 # 숫자만
data.corr(numeric_only=True)
```

Out[10]:

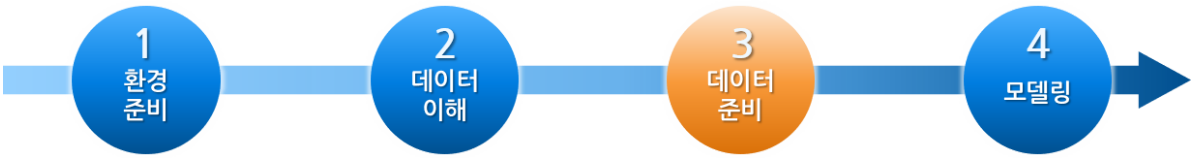
	Ozone	Solar.R	Wind	Temp	Month	Day
Ozone	1.000000	0.280068	-0.605478	0.683372	0.174197	0.004419
Solar.R	0.280068	1.000000	-0.056792	0.275840	-0.075301	-0.150275
Wind	-0.605478	-0.056792	1.000000	-0.457988	-0.178293	0.027181
Temp	0.683372	0.275840	-0.457988	1.000000	0.420947	-0.130593
Month	0.174197	-0.075301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

In [28]:
```
# 상관관계 시각화
plt.figure(figsize=(6, 3))
sns.heatmap(data.corr(numeric_only=True), annot=True, cbar=False, fmt='.3f', square=True, cmap
plt.show()
```



3.데이터 준비

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



1) 결측치 처리

- 결측치가 있으면 제거하거나 적절한 값으로 채웁니다.

In [29]:
```
# 결측치 확인
data.isna().sum()
```

Out[29]:
```
Ozone      0
Solar.R    7
Wind       0
Temp       0
Month      0
Day        0
dtype: int64
```

In [30]:
```
# 전날 값으로 결측치 채우기
data.fillna(method='ffill', inplace=True)

# 확인
data.isna().sum()
```

Out[30]:
```
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 [31]:
```
# 변수 제거
drop_cols = ['Month', 'Day']
data.drop(drop_cols, axis=1, inplace=True)


# 확인
data.head()
```

Out[31]:

	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 [32]:
```
# target 확인
target = 'Ozone'

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

In [36]:
```
# x 확인
x.head()
```

Out[36]:

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

In [37]:
```
# y 확인
y.head()
```

Out[37]:
```
0    41
1    36
2    12
3    18
4    19
Name: Ozone, dtype: int64
```

4) 학습용, 평가용 데이터 분리

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

In [41]:
```
# 모듈 불러오기
from sklearn.model_selection import train_test_split # 데이터를 무자위로 섞음

# 7:3으로 분리
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1, shuff
```

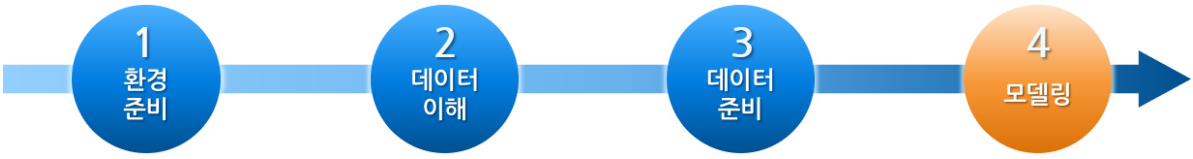
In [42]:
```
# 확인
x_train.head()
```

Out[42]:

	Solar.R	Wind	Temp
132	259.0	9.7	73
73	175.0	14.9	81
18	322.0	11.5	68
48	37.0	9.2	65
4	313.0	14.3	56

4.모델링

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



1) 모델링

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

In [43]:
```
# 1단계: 불러오기
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error # 평가는 metrics에서 다 불러올 수 있다.
```

In [71]:
```
# 2단계: 선언하기
model = LinearRegression()
```

In [72]:
```
# 3단계: 학습하기
model.fit(x_train, y_train)
```

Out[72]:
```
LinearRegression()
```

In [73]:
```
# 4단계: 예측하기
y_pred = model.predict(x_test)
```

In [74]:
```
# 실제값, 예측값 비교
print(y_test.values[:10])
print(y_pred[:10])
```

```
[24 18 97 47 34 22 66 18 69 27]
[13.84003067  5.82919112 81.93563027 58.41267418 50.86150737 31.52971121
 66.8083547  -8.56411529 50.2136544  39.13346172]
```

In [75]:
```
# 5단계: 평가하기
# 평균 절대 오차 # 값이 낮을 수록 좋은 모델
print('MAE:', mean_absolute_error(y_test, y_pred))
```

```
MAE: 13.976843190385708
```

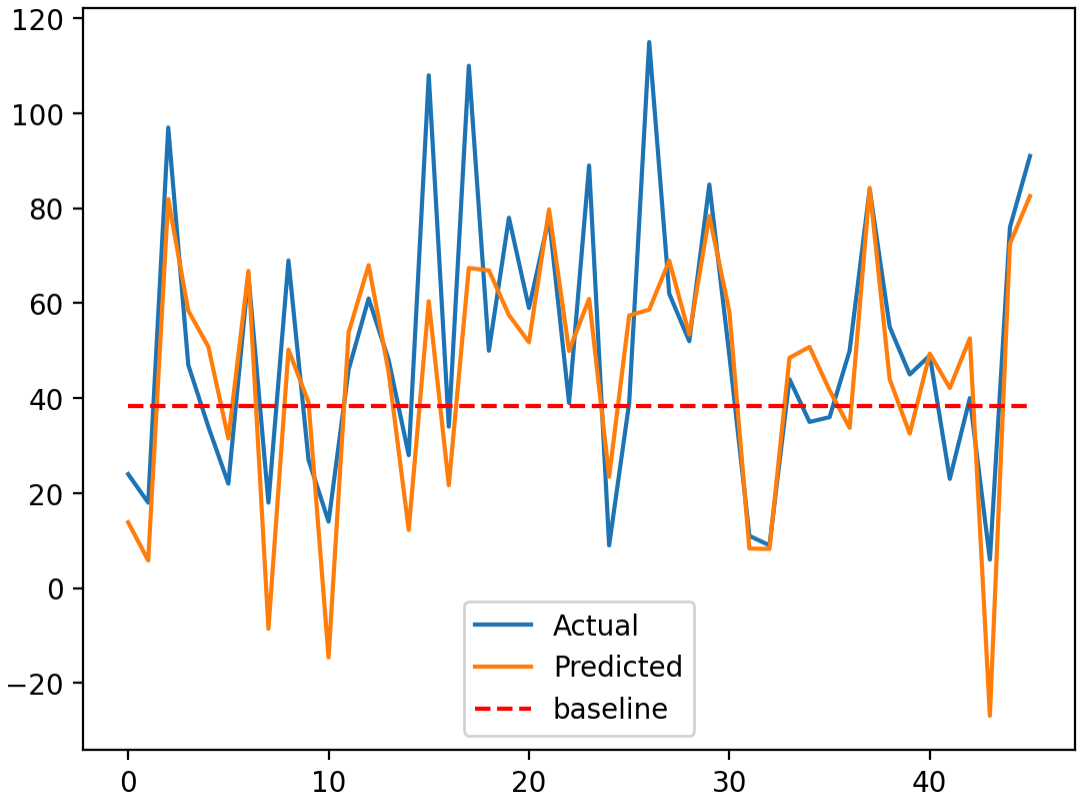
In [76]:
```
# 기준(평균) 모델 성능 평가
mean_ozone = y_train.mean()
y_base= np.array([mean_ozone] * len(y_test))
```

In [77]:
```
# 5단계: 평가하기
# 평균 절대 오차 # 값이 낮을 수록 좋은 모델
print('MAE:', mean_absolute_error(y_test, y_base))
```

```
MAE: 23.823852092645264
```

2) 결과 시각화

In [78]:
```
plt.plot(y_test.values, label='Actual')
plt.plot(y_pred, label='Predicted')
plt.plot(y_base, label='baseline', color='r', linestyle='--')
plt.legend()
plt.show()
```



In [ ]:
```

```