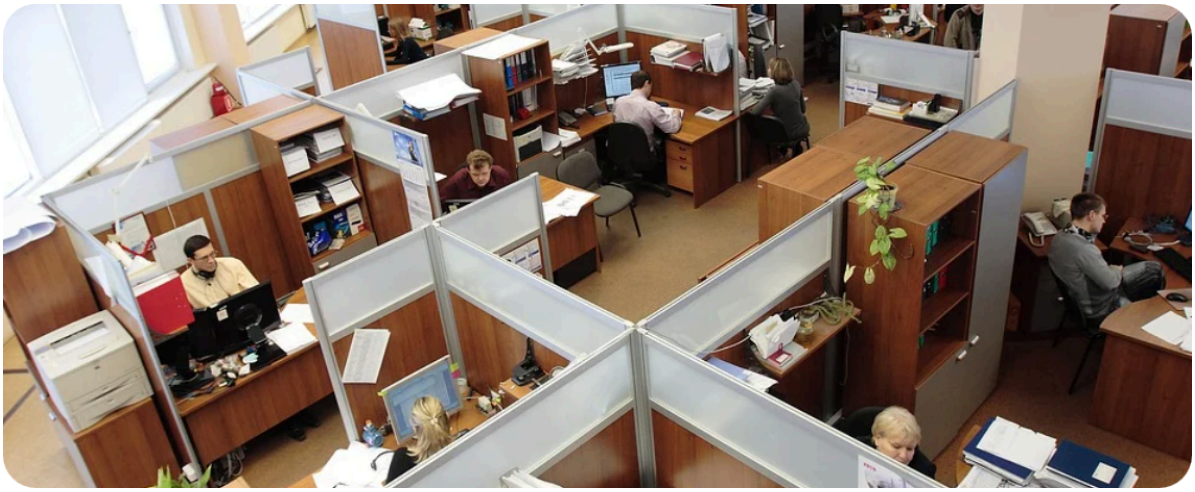
Machine Learning with Python

Life is too short, You need Python



실습 내용

- Attrition 데이터로 모델링합니다.
- KNN 알고리즘으로 모델링합니다.

1.환경 준비

- 기본 라이브러리와 대상 데이터를 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 [14]:
```
# 데이터 읽어오기
path = 'https://raw.githubusercontent.com/jangrae/csv/master/Attrition_simple2.csv'
data = pd.read_csv(path)
```

2.데이터 이해

- 분석할 데이터를 충분히 이해할 수 있도록 다양한 탐색 과정을 수행합니다.

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

Out[3]:

	Attrition	Age	DistanceFromHome	EmployeeNumber	Gender	JobSatisfaction	MaritalStatus	Mon
0	0	33	7	817	Male	3	Married	
1	0	35	18	1412	Male	4	Single	
2	0	42	6	1911	Male	1	Married	
3	0	46	2	1204	Female	1	Married	
4	1	22	4	593	Male	3	Single	

데이터 설명

- Attrition: 이직 여부 (1: 이직, 0: 잔류)
- Age: 나이
- DistanceFromHome: 집-직장 거리 (단위: 마일)
- EmployeeNumber: 사번
- Gender: 성별 (Male, Female)
- JobSatisfaction: 직무 만족도(1: Low, 2: Medium, 3: High, 4: Very High)
- MaritalStatus: 결혼 상태 (Single, Married, Divorced)
- MonthlyIncome: 월급 (단위: 달러)
- OverTime: 야근 여부 (Yes, No)
- PercentSalaryHike: 전년 대비 급여 인상율(단위: %)
- TotalWorkingYears: 총 경력 연수

In [4]:
```
# 기술통계 확인
data.describe()
```

Out[4]:

	Attrition	Age	DistanceFromHome	EmployeeNumber	JobSatisfaction	MonthlyIncom
count	1196.000000	1196.00000	1196.000000	1196.000000	1196.000000	1196.00000
mean	0.163043	36.94398	9.258361	1035.629599	2.716555	6520.10451
std	0.369560	9.09270	8.166016	604.340130	1.110962	4665.90225
min	0.000000	18.00000	1.000000	1.000000	1.000000	1009.00000
25%	0.000000	30.00000	2.000000	507.750000	2.000000	2928.25000
50%	0.000000	36.00000	7.000000	1028.000000	3.000000	4973.50000
75%	0.000000	43.00000	14.000000	1581.250000	4.000000	8420.50000
max	1.000000	60.00000	29.000000	2068.000000	4.000000	19999.00000

In [5]:
```
# NaN 값 확인
data.isnull().sum()
```

Out[5]:
```
Attrition            0
Age                  0
DistanceFromHome     0
EmployeeNumber       0
Gender               0
JobSatisfaction      0
MaritalStatus        0
MonthlyIncome        0
OverTime             0
PercentSalaryHike    0
TotalWorkingYears    0
dtype: int64
```

In [6]:
```
# 상관관계 확인
data.corr(numeric_only=True)
```

Out[6]:

	Attrition	Age	DistanceFromHome	EmployeeNumber	JobSatisfaction	Mon
Attrition	1.000000	-0.167866	0.081973	-0.008707	-0.078936	
Age	-0.167866	1.000000	-0.010917	-0.023786	-0.012425	
DistanceFromHome	0.081973	-0.010917	1.000000	0.054948	-0.021623	
EmployeeNumber	-0.008707	-0.023786	0.054948	1.000000	-0.022863	
JobSatisfaction	-0.078936	-0.012425	-0.021623	-0.022863	1.000000	
MonthlyIncome	-0.163572	0.490107	-0.012803	-0.014032	-0.025082	
PercentSalaryHike	-0.000048	-0.008303	0.052348	-0.009514	0.030811	
TotalWorkingYears	-0.182162	0.674331	0.002606	-0.016317	-0.039380	

3.데이터 준비

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

1) 변수 제거

- 제거 대상 변수: EmployeeNumber

In [15]:
```
# 제거 대상: EmployeeNumber
drop_cols = ['EmployeeNumber']

# 변수 제거
data.drop(drop_cols, axis=1, inplace=True)

# 확인
data.head()
```

Out[15]:

	Attrition	Age	DistanceFromHome	Gender	JobSatisfaction	MaritalStatus	MonthlyIncome	OverTi
0	0	33	7	Male	3	Married	11691	
1	0	35	18	Male	4	Single	9362	
2	0	42	6	Male	1	Married	13348	
3	0	46	2	Female	1	Married	17048	
4	1	22	4	Male	3	Single	3894	

2) x, y 분리

In [16]:
```
# target 확인
target = 'Attrition'

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

3) 가변수화

In [24]:
```
# 가변수화 대상: Gender, JobSatisfaction, MaritalStatus, OverTime
dumm_cols = ['Gender', 'JobSatisfaction', 'MaritalStatus', 'OverTime']

# 가변수화
x = pd.get_dummies(x, columns=dumm_cols, drop_first=True, dtype=int)

# 확인
x.head()
```

Out[24]:

	Age	DistanceFromHome	MonthlyIncome	PercentSalaryHike	TotalWorkingYears	Gender_Male	Job
0	33	7	11691	11	14	1	
1	35	18	9362	11	10	1	
2	42	6	13348	13	18	1	
3	46	2	17048	23	28	0	
4	22	4	3894	16	4	1	

4) 학습용, 평가용 데이터 분리

In [25]:
```
# 모듈 불러오기
from sklearn.model_selection import train_test_split

# 데이터 분리
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1)
```

5) 정규화

In [39]:
```
# 모듈 불러오기
from sklearn.preprocessing import MinMaxScaler

# 정규화
scaler = MinMaxScaler()
scaler.fit(x_train)
x_train_s = scaler.transform(x_train)
x_test_s = scaler.transform(x_test)
```

4.모델링

- 본격적으로 모델을 선언하고 학습하고 평가하는 과정을 진행합니다.

In [31]:
```
# 1단계: 불러오기
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import mean_absolute_error, r2_score
```

In [32]:
```
# 2단계: 선언하기
model = KNeighborsClassifier()
```

In [33]:
```
# 3단계: 학습하기
model.fit(x_train, y_train)
```

Out[33]:
```
KNeighborsClassifier()
```

In [34]:
```
# 4단계: 예측하기
y_pred = model.predict(x_test)
```

In [35]:
```
# 5단계: 평가하기
print('MAE:', mean_absolute_error(y_test, y_pred))
print('R2:', r2_score(y_test, y_pred))
```

```
MAE: 0.19220055710306408
R2: -0.39949152542372857
```

In [40]:
```
# 정규화 후
# 4단계: 예측하기
y_pred = model.predict(x_test_s)
```

In [41]:
```
# 5단계: 평가하기
print('MAE:', mean_absolute_error(y_test, y_pred))
print('R2:', r2_score(y_test, y_pred))
```

```
MAE: 0.16434540389972144
R2: -0.19666666666666655
```

In [ ]: