사이킷런 설치

In [1]:
```
!pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in /Users/youngjinseo/anaconda3/lib/python3.10/site-packages (1.2.1)
Requirement already satisfied: numpy>=1.17.3 in /Users/youngjinseo/anaconda3/lib/python3.10/site-packages (from scikit-learn) (1.23.5)
Requirement already satisfied: joblib>=1.1.1 in /Users/youngjinseo/anaconda3/lib/python3.10/site-packages (from scikit-learn) (1.1.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Users/youngjinseo/anaconda3/lib/python3.10/site-packages (from scikit-learn) (2.2.0)
Requirement already satisfied: scipy>=1.3.2 in /Users/youngjinseo/anaconda3/lib/python3.10/site-packages (from scikit-learn) (1.10.0)
```

라이브러리 불러오기

In [9]:
```
#데이터프레임
import pandas as pd

#데이터 시각화
import seaborn as sns

#전처리 스케일링
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import MaxAbsScaler
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import LabelEncoder
```

In [22]:
```
#titanic 데이터셋 불러오기
df = sns.load_dataset("titanic")
df.head()
```

Out[22]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

In [23]:
```
#age,fare

df1 = df[['age','fare']]
df2.head(4)
```

Out[23]:

	age	fare
0	22.0	7.2500
1	38.0	71.2833
2	26.0	7.9250
3	35.0	53.1000

표준화 (Standardization) : 많이 활용함

- 변수 각각의 평균을 0, 분산을 1로 만들어주는 스케일링 기법입니다.
- 표준화가 적용된 변수는 가우시안 정규분포를 가진 값으로 변환됩니다.
- 최귀보다 분류에 유용합니다.

StandardScaler

In [24]:
```
# StandardScaler 객체를 생성함
stand = StandardScaler()

# fit_transform()을 사용해서 학습과 스케일링을 한 번에 적용
stand_train = stand.fit_transform(df2)


# 완료된 데이터를 데이터프레임 형태로 변환
stand_train = pd.DataFrame(stand_train,columns = ['age','fare'])
stand_train.head(5)
```

Out[24]:

	age	fare
0	-0.530377	-0.502445
1	0.571831	0.786845
2	0.254825	-0.488854
3	0.365167	0.420730
4	0.365167	-0.486337

정규화 (Normalization)

- 일반적으로 서로 다른 변수의 크기를 통일하기 위해 크기를 변환해주는 개념입니다.
- 데이터를 0과 1 사이의 값으로 변환합니다. 최소값을 0, 최대값을 1로 맞춥니다.
- 분류보다 회귀에 유용합니다.

MinMaxScaler(), MaxAbsScaler()

In [28]:
```
#MinMaxScaler
minmax = MinMaxScaler()

# fit_transform()을 사용해서 학습과 스케일링을 한 번에 적용
minmax_train = minmax.fit_transform(df2)

# 완료된 데이터를 데이터프레임 형태로 변환
minmax_train = pd.DataFrame(minmax_train, columns = ['age','fare'])
minmax_train.head(8)
```

Out[28]:

	age	fare
0	0.271174	0.014151
1	0.472229	0.139136
2	0.321438	0.015469
3	0.434531	0.103644
4	0.434531	0.015713
5	NaN	0.016510
6	0.673285	0.101229
7	0.019854	0.041136

In [29]:
```
#MaxAbsScaler()
maxabs = MaxAbsScaler()


# fit_transform()을 사용해서 학습과 스케일링을 한 번에 적용
max_train =maxabs.fit_transform(df2)

# 완료된 데이터를 데이터프레임 형태로 변환
max_train = pd.DataFrame(max_train,columns = ['age','fare'])
max_train.head(8)
```

Out[29]:

	age	fare
0	0.2750	0.014151
1	0.4750	0.139136
2	0.3250	0.015469
3	0.4375	0.103644
4	0.4375	0.015713
5	NaN	0.016510
6	0.6750	0.101229
7	0.0250	0.041136

로버스트 (Robust)

- 중앙값과 사분위수를 사용해 이상치에 덜 민감하게 데이터를 스케일링합니다.
- 평균과 분산 대신에 중간 값과 사분위 값을 사용합니다. (중간 값은 정렬시 중간에 있는 값을 의미하고, 사분위값은 1/4, 3/4에 위치한 값을 의미합니다.)
- RobustScaler를 사용하면 모든 변수들이 같은 스케일을 갖게 되며, StandardScaler에 비해 스케일링 결과가 더 넓은 범위로 분포하게 됩니다.따라서 StandardScaler에 비해 이상치의 영향이 적어진다는 장점이 있습니다.

In [31]:
```
#RobustScaler()
robust = RobustScaler()


# fit_transform()을 사용해서 학습과 스케일링을 한 번에 적용
robust_train = robust.fit_transform(df2)


# 완료된 데이터를 데이터프레임 형태로 변환
robust_train = pd.DataFrame(robust_train,columns = ['age','fare'])
robust_train.head(7)
```

Out[31]:

	age	fare
0	-0.335664	-0.312011
1	0.559441	2.461242
2	0.111888	-0.282777
3	0.391608	1.673732
4	0.391608	-0.277363
5	NaN	-0.259680
6	1.454545	1.620136

레이블 인코딩(Label Encoding)

- 따라서 문자열 값을 인코딩해서 숫자형으로 변환합니다.
- 예를 들어, '고양이', '개', '사자'라는 세 가지 레이블이 있다면, 이를 각각 0, 1, 2로 변환하는 방식입니다.

In [37]:
```
df3 = df['class']
```

In [39]:
```
#Label Encoding
labelenco = LabelEncoder()

# fit_transform()을 사용해서 학습과 스케일링을 한 번에 적용
label_train = labelenco.fit_transform(df3)

# 완료된 데이터를 데이터프레임 형태로 변환
label_train = pd.DataFrame(label_train, columns = ['class'])
label_train.head(5)
```

Out[39]:

	class
0	2
1	0
2	2
3	0
4	2

주의해야 할 점

학습 데이터로 fit()과 transform()을 적용하면, 테스트 데이터로는 다시 fit()을 수행하지 않고 학습 데이터로 fit()을 수행한 결과를 이용해 transform() 변환을 적용해야 한다는 것입니다.

즉 테스트 데이터에 다시 fit()을 적용해서는 안 되며, 학습 데이터로 이미 fit()이 적용된 Scaler 객체를 이용해 transform()으로 변환해야 합니다.

문제를 한번 풀어볼까요?

In [45]:
```
mf = sns.load_dataset('iris')
mf
```

Out[45]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [42]:
```
#LabelEncoder

species = mf['species']

#클래인덕기
labelenco = LabelEncoder()

#변환하기
label_train = labelenco.fit_transform(species)

#dataframe
label_train = pd.DataFrame(label_train, columns = ['species'])
label_train.head(10)
```

Out[42]:

	species
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

In [44]:
```
#StandardScaler

df3 = mf[['petal_length','petal_width']]

stand = StandardScaler()

stand_train = stand.fit_transform(df3)

stand_train = pd.DataFrame(stand_train,columns = ['petal_length','petal_width'])
stand_train.head(10)
```

Out[44]:

	petal_length	petal_width
0	-1.340227	-1.315444
1	-1.340227	-1.315444
2	-1.397064	-1.315444
3	-1.283389	-1.315444
4	-1.340227	-1.315444
5	-1.169714	-1.052180
6	-1.340227	-1.183812
7	-1.283389	-1.315444
8	-1.340227	-1.315444
9	-1.283389	-1.447076

In [48]:
```
#MaxAbsScaler

maxabs = MaxAbsScaler()

max_train = maxabs.fit_transform(df3)

max_train = pd.DataFrame(max_train,columns = ['petal_length','petal_width'])
max_train.head(10)
```

Out[48]:

	petal_length	petal_width
0	0.202899	0.08
1	0.202899	0.08
2	0.188406	0.08
3	0.217391	0.08
4	0.202899	0.08
5	0.246377	0.16
6	0.202899	0.12
7	0.217391	0.08
8	0.202899	0.08
9	0.217391	0.04

In [ ]:
```
#RobustScaler
robust = RobustScaler()

robust_train = robust.fit_transform()

robust_train = pd.DataFrame(robust_train,columns = ['petal_length','petal_width'])
```