유방암 여부 로지스틱 회귀 예제

라이브러리 및 패키지 Import

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_20636\1062926837.py:1: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better interoperability with other lib
raries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

데이터셋 불러오기

In [2]:
```python
breast_cancer = load_breast_cancer()
```

In [3]:
```python
# 로드한 전체 데이터에 key 값을 출력
print(breast_cancer.keys())
# 전체 데이터 중 data에 대한 전체 행, 열 길이를 출력
print(breast_cancer.data.shape)
# 데이터 컬럼 이름을 출력
print(breast_cancer.feature_names)
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
(569, 30)
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
```

In [4]:
```python
print(breast_cancer.DESCR)
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

:Number of Instances: 569

:Number of Attributes: 30 numeric, predictive attributes and the class

:Attribute Information:
    - radius (mean of distances from center to points on the perimeter)
    - texture (standard deviation of gray-scale values)
    - perimeter
    - area
    - smoothness (local variation in radius lengths)
    - compactness (perimeter^2 / area - 1.0)
    - concavity (severity of concave portions of the contour)
    - concave points (number of concave portions of the contour)
    - symmetry
    - fractal dimension ("coastline approximation" - 1)

    The mean, standard error, and "worst" or largest (mean of the three
    worst/largest values) of these features were computed for each image,
    resulting in 30 features.  For instance, field 0 is Mean Radius, field
    10 is Radius SE, field 20 is Worst Radius.

    - class:
            - WDBC-Malignant
            - WDBC-Benign

:Summary Statistics:

===================================== ====== ======
```

	Min	Max
radius (mean):	6.981	28.11
texture (mean):	9.71	39.28
perimeter (mean):	43.79	188.5
area (mean):	143.5	2501.0
smoothness (mean):	0.053	0.163
compactness (mean):	0.019	0.345
concavity (mean):	0.0	0.427
concave points (mean):	0.0	0.201
symmetry (mean):	0.106	0.304
fractal dimension (mean):	0.05	0.097
radius (standard error):	0.112	2.873
texture (standard error):	0.36	4.885
perimeter (standard error):	0.757	21.98
area (standard error):	6.802	542.2
smoothness (standard error):	0.002	0.031
compactness (standard error):	0.002	0.135
concavity (standard error):	0.0	0.396
concave points (standard error):	0.0	0.053
symmetry (standard error):	0.008	0.079
fractal dimension (standard error):	0.001	0.03
radius (worst):	7.93	36.04
texture (worst):	12.02	49.54
perimeter (worst):	50.41	251.2
area (worst):	185.2	4254.0
smoothness (worst):	0.071	0.223
compactness (worst):	0.027	1.058
concavity (worst):	0.0	1.252
concave points (worst):	0.0	0.291
symmetry (worst):	0.156	0.664
fractal dimension (worst):	0.055	0.208

:Missing Attribute Values: None

:Class Distribution: 212 - Malignant, 357 - Benign

:Creator: Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

:Donor: Nick Street

:Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle
aspirate (FNA) of a breast mass. They describe
characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using
Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree
Construction Via Linear Programming." Proceedings of the 4th
Midwest Artificial Intelligence and Cognitive Science Society,
pp. 97-101, 1992], a classification method which uses linear
programming to construct a decision tree. Relevant features
were selected using an exhaustive search in the space of 1-4
features and 1-3 separating planes.

The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

```
ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/
```

|details-start|
References
|details-split|

- W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction
  for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on
  Electronic Imaging: Science and Technology, volume 1905, pages 861-870,
  San Jose, CA, 1993.
- O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and
  prognosis via linear programming. Operations Research, 43(4), pages 570-577,
  July-August 1995.
- W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques
  to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994)
  163-171.

|details-end|

데이터 전처리

In [5]:
```
df = pd.DataFrame(data = breast_cancer.data, columns = breast_cancer.feature_names)
df = df.iloc[:, :10]
```

In [6]:
```
df["label"] = breast_cancer.target
df.columns = [ col.replace(" ", "_") for col in df.columns]
df.head()
```

Out[6]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	mean_compactness	mean_concavity	mean_concave_points
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430

In [7]:
```
# Step1) train / test 으로 나누기
train, test = train_test_split(df, test_size=0.15, random_state=1)

# Step2) train을 다시 train/validation 으로 나누기
train, val = train_test_split(train, test_size = 0.18, random_state=1)

# 확인
train.shape, val.shape, test.shape
```

Out[7]:
```
((396, 11), (87, 11), (86, 11))
```

In [8]:
```
# feature/taraget 설정
feature = train.columns[:-1]
target = "label"

# train 데이터셋
X_train = train[feature]
y_train = train[target]

# validation 데이터셋
X_val = val[feature]
y_val = val[target]

# test 데이터셋
X_test = test[feature]
y_test = test[target]

# 확인
print("feature Matrix: ", X_train.shape, X_val.shape, X_test.shape)
print("target vector: ", y_train.shape, y_val.shape, y_test.shape)
```

```
feature Matrix:  (396, 10) (87, 10) (86, 10)
target vector:  (396,) (87,) (86,)
```

In [9]:
```
# 스케일러 생성
scaler = StandardScaler()

# 스케일 조정
X_train_sclaed = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)

# 일부만 확인
X_train_sclaed.T[0].mean(), X_train_sclaed.std()
```

Out[9]:
```
(-2.444733528972567e-16, 1.0)
```

로지스틱 회귀 모델 학습

In [10]:
```
# 모델 생성 및 학습 시키기
logistic = LogisticRegression()
logistic.fit(X_train_sclaed, y_train)
```

Out[10]:
```
▾ LogisticRegression
LogisticRegression()
```

In [11]:
```
# Validation set 결과 확인
print("validation 데이터셋 정확도")
logistic.score(X_val_scaled, y_val)
```

```
validation 데이터셋 정확도
0.9310344827586207
```

Out[11]:

In [12]:
```
# 각 Feature의 계수 확인(영향도 분석)
logistic.coef_
```

Out[12]:
```
array([[-0.94054808, -1.31667477, -0.81743154, -1.03033924, -0.90780749,
         0.08604782, -1.17240721, -1.42903504, -0.40733798,  0.4668457 ]])
```

Test set 결과 확인

In [13]:
```
X_test_scaled = scaler.transform(X_test)

# 결과 확인
print("test 데이터셋 정확도")
logistic.score(X_test_scaled, y_test)
```

```
test 데이터셋 정확도
```

Out[13]:
```
0.9534883720930233
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js