# 위스콘신 유방암 데이터의 스태킹 모델 만들기

## 학습 목표

- 스태킹 모델을 구현해 본다.

## 목차



## 01 라이브러리 설치 및 불러오기

목차로 이동하기

In [53]:
```
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_breast_cancer

from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

import pandas as pd
import numpy as np

import warnings
warnings.filterwarnings(action='ignore')
```

## 02 데이터 준비 및 기본 모델 지정

목차로 이동하기

### 데이터 불러오기

In [54]:
```
cancer = load_breast_cancer()

cancer_df = pd.DataFrame(cancer.data, columns=cancer.feature_names)
cancer_df['target'] = cancer.target
cancer_df
```

Out[54]:

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 | 0.30010 | 0.14710 | 0.241 |
| **1** | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | 0.08690 | 0.07017 | 0.181 |
| **2** | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | 0.19740 | 0.12790 | 0.206 |
| **3** | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 | 0.24140 | 0.10520 | 0.259 |
| **4** | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 | 0.19800 | 0.10430 | 0.180 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **564** | 21.56 | 22.39 | 142.00 | 1479.0 | 0.11100 | 0.11590 | 0.24390 | 0.13890 | 0.172 |
| **565** | 20.13 | 28.25 | 131.20 | 1261.0 | 0.09780 | 0.10340 | 0.14400 | 0.09791 | 0.175 |
| **566** | 16.60 | 28.08 | 108.30 | 858.1 | 0.08455 | 0.10230 | 0.09251 | 0.05302 | 0.159 |
| **567** | 20.60 | 29.33 | 140.10 | 1265.0 | 0.11780 | 0.27700 | 0.35140 | 0.15200 | 0.239 |
| **568** | 7.76 | 24.54 | 47.92 | 181.0 | 0.05263 | 0.04362 | 0.00000 | 0.00000 | 0.158 |

569 rows × 31 columns

## 데이터 준비, 모델 지정

In [55]:
```
# 데이터를 준비합니다.
X = cancer_df.drop(['target'], axis=1)
y = cancer_df['target']

X.shape, y.shape
```

Out[55]:
```
((569, 30), (569,))
```

In [59]:
```
# 기본 모델을 초기화합니다.
rf_model = RandomForestClassifier()
lr_model = LogisticRegression()
knn_model = KNeighborsClassifier()

# 데이터를 훈련 세트와 테스트 세트로 나눕니다.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=42)

### 학습용 세트를 학습과 검증 세트로 나눈다.
X_tr, X_val, y_tr, y_val = train_test_split(X_train, y_train,
                                            test_size=0.2, random_state=42)
```

## 모델 훈련 및 각각의 모델 예측 결과 저장

In [60]:
```
# 기본 모델을 훈련시키고 예측 결과를 저장합니다.
rf_model.fit(X_tr, y_tr)
rf_pred_val = rf_model.predict(X_val)

lr_model.fit(X_tr, y_tr)
lr_pred_val = lr_model.predict(X_val)

knn_model.fit(X_tr, y_tr)
knn_pred_val = knn_model.predict(X_val)
```

## 03 메타 모델 지정 및 데이터 준비

목차로 이동하기

In [61]:
```
# 예측 결과를 기반으로 메타 모델을 훈련시킵니다.
meta_features_pred_val = [rf_pred_val, lr_pred_val, knn_pred_val]
meta_X = np.array(meta_features_pred_val).T
meta_X[0:3]
```

Out[61]:
```
array([[0, 0, 0],
       [1, 1, 1],
       [1, 1, 1]])
```

In [62]:
```
meta_y = y_val
print(type(meta_X), type(meta_y))
meta_X.shape, meta_y.shape
```

Out[62]:
```
<class 'numpy.ndarray'> <class 'pandas.core.series.Series'>
((91, 3), (91,))
```

- 메타 모델의 행은 검증용 데이터 행과 같다.
- 메타 모델의 열은 **모델의 개수**와 열이 같다.
- 입력(meta_X는 각 모델의 예측 결과)
- 출력(meta_y는 검증용 y의 값)

In [63]:
```
meta_model = RandomForestClassifier()
meta_model.fit(meta_X, meta_y)
```

Out[63]:
```
RandomForestClassifier()
```

### 메타 모델에 넣을 새로운 데이터 셋을 만든다.

- 처음에 준비한 X_test데이터 셋을 활용.

In [66]:
```
rf_pred_test = rf_model.predict(X_test)
lr_pred_test = lr_model.predict(X_test)
knn_pred_test = knn_model.predict(X_test)
```

In [67]:
```
# 테스트 데이터에 대한 예측을 생성합니다.
test_meta_features = [rf_pred_test, lr_pred_test, knn_pred_test]
test_meta_X = np.array(test_meta_features).T

# 메타 모델을 사용하여 테스트 데이터에 대한 최종 예측을 생성합니다.
final_pred = meta_model.predict(test_meta_X)
```

### 최종 예측 결과와 테스트 y값을 비교하여 결과 확인

In [68]:
```
# 최종 예측의 정확도를 평가합니다.
accuracy = accuracy_score(y_test, final_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9649122807017544
```