AdaBoost

Adaptive Boost

초기 모형을 약한 모형으로 설정하며 매 스탭마다 가중치를 이용하여 이전 모형의 약점을 보완하는 방식으로 새로운 모형을 순차적으로 학습하고 최종적으로 이들을 선형 결합하여 얻어진 모형을 생성하는 알고리즘

#01. 패키지 가져오기

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

from matplotlib import pyplot as plt
from pandas import read_excel
from sklearn.ensemble import AdaBoostClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.metrics import roc_curve, roc_auc_score, auc, RocCurveDisplay
from sklearn.model_selection import GridSearchCV
from imblearn.over_sampling import SMOTE
from sklearn.tree import DecisionTreeClassifier

from sklearnex import patch_sklearn
from daal4py.oneapi import sycl_context
patch_sklearn()
```

```
Intel(R) Extension for Scikit-learn* enabled (https://github.com/intel/scikit-learn-in
```

#02. 데이터 가져오기

```
origin = read_excel('https://data.hossam.kr/G02/breast_cancer.xlsx')
origin.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	me symme
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809

5 rows × 31 columns

#03. 데이터 전처리

독립/종속 변수 분리

```
x = origin.drop('target', axis=1)
y = origin['target']
x.shape, y.shape
```

```
((569, 30), (569,))
```

훈련, 검증 데이터 분리

```
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state = 123)
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

```
((426, 30), (143, 30), (426,), (143,))
```

데이터 불균형 처리

```
smote_sampler = SMOTE(sampling_strategy="minority", random_state=777)
x_sm, y_sm = smote_sampler.fit_resample(x_train, y_train)
print(x_sm.shape, y_sm.shape)

y_sm.value_counts().sort_index()
```

```
(536, 30) (536,)
```

```
0    268
1    268
Name: target, dtype: int64
```

#04. 훈련 모델 적합

부스팅에 사용할 학습 알고리즘 생성

```
dt = DecisionTreeClassifier(max_depth=2, min_samples_leaf=10,
                            random_state=123)
dt
```

▾ DecisionTreeClassifier

```
DecisionTreeClassifier(max_depth=2, min_samples_leaf=10, random_state=123)
```

AdaBoost 생성

하이퍼파라미터

파라미터	설명
base_estimator	학습에 사용하는 알고리즘 (default=`DecisionTreeClassifier(max_depth=1)`)
n_estimators	반복수 또는 base_estimator 개수(기본값=50)
learning_rate	학습을 진행할 때마다 적용하는 학습률 `0 ~ 1`의 값(기본값=0.1)
algorithm	`SAMME` - 이산 부스팅 알고리즘, `SAMME.R` - 부스팅 알고리즘(기본값=`SAMME.R`)

n_estimators 를 늘린다면 생성되는 약한 학습기의 수는 늘어난다. 하지만 이 여러 학습기들의 decision boundary가 많아지면서 모델이 복잡해진다.

learning_rate 을 줄인다면, 가중치의 갱신 변동폭이 감소해서, 여러 학습기들의 decision boundary의 차이가 줄어든다.

```
ada = AdaBoostClassifier(
        base_estimator=dt,  # 훈련에 사용할 학습 모델
        n_estimators=5,     # 학습 모델의 개수(또는 반복 횟수)
        learning_rate=0.1,  # 학습률
        random_state=123)

ada.fit(x_sm, y_sm)

print("훈련 정확도: ", ada.score(x_sm, y_sm))

y_pred = ada.predict(x_test)
print("테스트 정확도: ", accuracy_score(y_test, y_pred))
```

```
훈련 정확도:  0.9682835820895522
테스트 정확도:  0.972027972027972
```

분류 보고서

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.98      0.96        54
           1       0.99      0.97      0.98        89

    accuracy                           0.97       143
   macro avg       0.97      0.97      0.97       143
weighted avg       0.97      0.97      0.97       143
```

ROC 곡선

각 클래스에 속할 확률에서 `1`에 속할 확률만 구함

```
score1 = ada.predict_proba(x_test)[:, 1]
score1[:5]
```

```
array([9.60005974e-01, 9.60005974e-01, 6.91078715e-05, 9.60005974e-01,
       6.91078715e-05])
```

ROC 점수 구하기

실제 Label과 Positive Label에 대한 예측 확률을 전달하여 roc 곡선 표현에 필요한 값들을 리턴받는다.

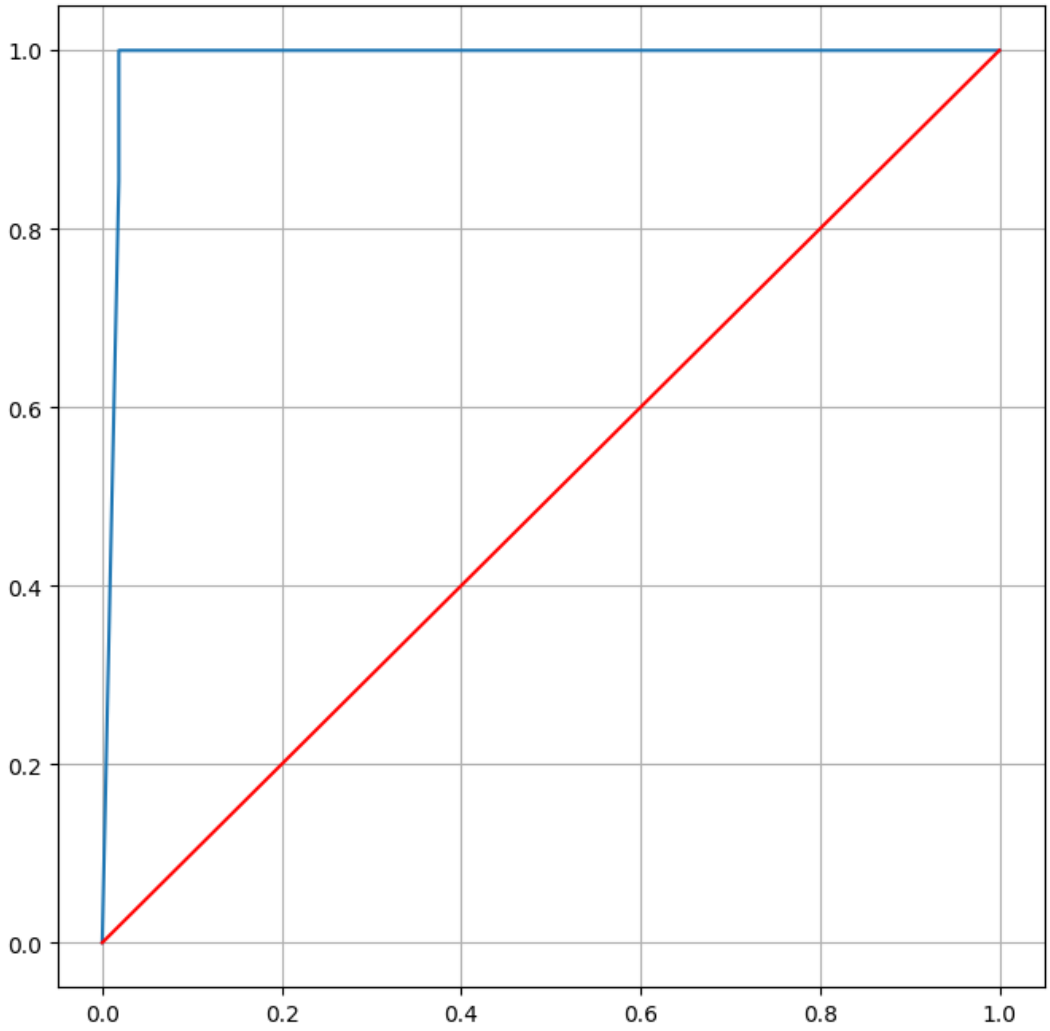
- 첫 번째 리턴값 : False Positive Rate(민감도)
- 두 번째 리턴값 : True Positive Rate(재현율)
- 세 번째 리턴값: 절단값(ROC커브 구현에 사용되지 않음)

```
fpr1, tpr1, cut1 = roc_curve(y_test, score1)
```

ROC 곡선 시각화

가운데 직선에 가까울 수록 분류 성능이 떨어지는 것이다.

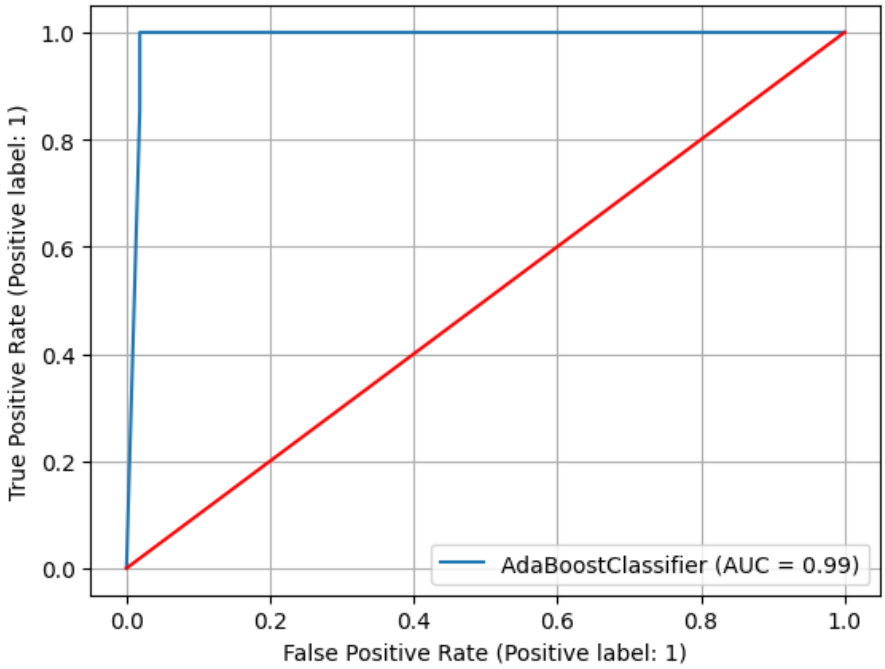
```
plt.figure(figsize=(8,8))
plt.plot(fpr1, tpr1)
plt.plot([0, 1], [0, 1], color='red')
plt.grid()
plt.show()
```



좀 더 이쁜 ROC 곡선

```
plt.figure(figsize=(8,8))
RocCurveDisplay.from_estimator(ada, x_test, y_test)
plt.plot([0, 1], [0, 1], color='red')
plt.grid()
plt.show()
plt.close()
```

```
<Figure size 800×800 with 0 Axes>
```



AUC 값 직접 계산하기

실제 Label과 Positive의 예측확률로 계산

```
print('roc_auc_score 함수 결과:', roc_auc_score(y_test, score1))
```

```
roc_auc_score 함수 결과: 0.9893882646691635
```

False Positive Rate와 True Positive Rate로 계산

```
print('auc 함수 결과:', auc(fpr1, tpr1))
```

```
auc 함수 결과: 0.9893882646691635
```

#05. 하이퍼파라미터 튜닝

```
dt = DecisionTreeClassifier(max_depth=2, min_samples_leaf=10,
                            random_state=123)

ada = AdaBoostClassifier(
        base_estimator=dt,
        random_state=123)

params = {
    "n_estimators": [3, 5, 10],
    "learning_rate": [0.1, 0.3, 0.5]
}

grid = GridSearchCV(ada, param_grid=params, cv=5, n_jobs=-1)
grid.fit(x_sm, y_sm)
```

```
print("최적의 하이퍼 파라미터: ", grid.best_params_)
print("최적의 모델 평균 성능(훈련데이터): ", grid.best_score_)

best_model = grid.best_estimator_
y_pred = best_model.predict(x_test)
print("최종 모델의 성능(테스트 데이터): ", accuracy_score(y_test, y_pred))
```

```
최적의 하이퍼 파라미터:  {'learning_rate': 0.5, 'n_estimators': 10}
최적의 모델 평균 성능(훈련데이터):  0.9571651090342679
최종 모델의 성능(테스트 데이터):  0.972027972027972
```

성능 검증

분류 보고서

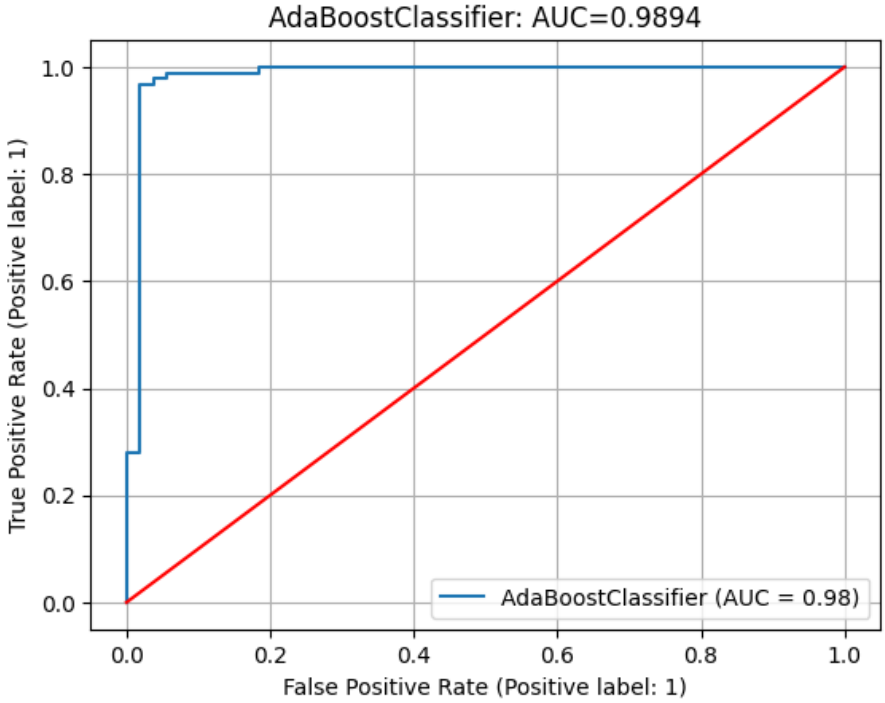
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.94      0.96        54
           1       0.97      0.99      0.98        89

    accuracy                           0.97       143
   macro avg       0.97      0.97      0.97       143
weighted avg       0.97      0.97      0.97       143
```

AUC, ROC 곡선

```
fpr1, tpr1, _ = roc_curve(y_test, score1)

plt.figure(figsize=(8,8))
RocCurveDisplay.from_estimator(best_model, x_test, y_test)
plt.plot([0, 1], [0, 1], color='red')
plt.title('AdaBoostClassifier: AUC={0:0.4f}'.format(auc(fpr1, tpr1)))
plt.grid()
plt.show()
plt.close()
```

```
<Figure size 800×800 with 0 Axes>
```


AdaBoostClassifier: AUC=0.9894
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
AdaBoostClassifier (AUC = 0.98)
0.0
0.2
0.4
0.6
0.8
1.0