SVM (1)

분류, 회귀, 이상치 감지에 사용되는 지도학습 알고리즘

#01. 패키지 참조

```
from pandas import DataFrame, read_excel
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score, cross_validate
```

#02. 데이터 가져오기

569개의 row, 31개의 column, 종속변수는 `[0, 1]`로 구분되어 있다.

30개의 독립변수를 통해 유방암 진단을 결정하는 데이터셋

```
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

#03. 데이터 전처리

전처리 과정에서 고민해 봐야 하는 단계

1. 결측치, 이상치 감지
2. 파생변수 생성여부 결정 및 수행

독립변수, 종속변수 분리

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

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

이 단계에서 고려해야 하는 단계

1. 표준화 적용 여부 (가급적 수행, before/after 결과 비교 권장)
2. 훈련 데이터와 검증 데이터 분할 (지도학습은 거의 필수라고 봐야 함)

```
scaler = StandardScaler()
std_x = scaler.fit_transform(x)
std_x[:5]
```

```
array([[ 1.09706398e+00, -2.07333501e+00,  1.26993369e+00,
         9.84374905e-01,  1.56846633e+00,  3.28351467e+00,
         2.65287398e+00,  2.53247522e+00,  2.21751501e+00,
         2.25574689e+00,  2.48973393e+00, -5.65265059e-01,
         2.83303087e+00,  2.48757756e+00, -2.14001647e-01,
         1.31686157e+00,  7.24026158e-01,  6.60819941e-01,
         1.14875667e+00,  9.07083081e-01,  1.88668963e+00,
        -1.35929347e+00,  2.30360062e+00,  2.00123749e+00,
         1.30768627e+00,  2.61666502e+00,  2.10952635e+00,
         2.29607613e+00,  2.75062224e+00,  1.93701461e+00],
       [ 1.82982061e+00, -3.53632408e-01,  1.68595471e+00,
         1.90870825e+00, -8.26962447e-01, -4.87071673e-01,
        -2.38458552e-02,  5.48144156e-01,  1.39236330e-03,
        -8.68652457e-01,  4.99254601e-01, -8.76243603e-01,
         2.63326966e-01,  7.42401948e-01, -6.05350847e-01,
        -6.92926270e-01, -4.40780058e-01,  2.60162067e-01,
        -8.05450380e-01, -9.94437403e-02,  1.80592744e+00,
        -3.69203222e-01,  1.53512599e+00,  1.89048899e+00,
        -3.75611957e-01, -4.30444219e-01, -1.46748968e-01,
         1.08708430e+00, -2.43889668e-01,  2.81189987e-01],
       [ 1.57988811e+00,  4.56186952e-01,  1.56650313e+00,
         1.55888363e+00,  9.42210440e-01,  1.05292554e+00,
         1.36347845e+00,  2.03723076e+00,  9.39684817e-01,
        -3.98007910e-01,  1.22867595e+00, -7.80083377e-01,
         8.50928301e-01,  1.18133606e+00, -2.97005012e-01,
         8.14973504e-01,  2.13076435e-01,  1.42482747e+00,
         2.37035535e-01,  2.93559404e-01,  1.51187025e+00,
        -2.39743838e-02,  1.34747521e+00,  1.45628455e+00,
         5.27407405e-01,  1.08293217e+00,  8.54973944e-01,
         1.95500035e+00,  1.15225500e+00,  2.01391209e-01],
       [-7.68909287e-01,  2.53732112e-01, -5.92687167e-01,
        -7.64463792e-01,  3.28355348e+00,  3.40290899e+00,
         1.91589718e+00,  1.45170736e+00,  2.86738293e+00,
         4.91091929e+00,  3.26373441e-01, -1.10409044e-01,
         2.86593405e-01, -2.88378148e-01,  6.89701660e-01,
         2.74428041e+00,  8.19518384e-01,  1.11500701e+00,
         4.73268037e+00,  2.04751088e+00, -2.81464464e-01,
         1.33984094e-01, -2.49939304e-01, -5.50021228e-01,
         3.39427470e+00,  3.89339743e+00,  1.98958826e+00,
         2.17578601e+00,  6.04604135e+00,  4.93501034e+00],
       [ 1.75029663e+00, -1.15181643e+00,  1.77657315e+00,
         1.82622928e+00,  2.80371830e-01,  5.39340452e-01,
```

```
        1.37101143e+00,  1.42849277e+00, -9.56046689e-03,
       -5.62449981e-01,  1.27054278e+00, -7.90243702e-01,
        1.27318941e+00,  1.19035676e+00,  1.48306716e+00,
       -4.85198799e-02,  8.28470780e-01,  1.14420474e+00,
       -3.61092272e-01,  4.99328134e-01,  1.29857524e+00,
       -1.46677038e+00,  1.33853946e+00,  1.22072425e+00,
        2.20556166e-01, -3.13394511e-01,  6.13178758e-01,
        7.29259257e-01, -8.68352984e-01, -3.97099619e-01]])
```

#04. 학습모델 구현

이 단계에서 표준화 적용 전후를 비교

표준화 적용 전

```
svc = SVC(random_state=777)
scores = cross_val_score(svc, x, y, cv=5)
print(scores)
print("교차검증 평균: ", scores.mean())

score_df = DataFrame(cross_validate(svc, x, y, cv=5))
score_df
```

```
[0.85087719 0.89473684 0.92982456 0.94736842 0.9380531 ]
교차검증 평균:  0.9121720229777983
```

	fit_time	score_time	test_score
0	0.005024	0.003015	0.850877
1	0.006060	0.003025	0.894737
2	0.005999	0.003000	0.929825
3	0.007974	0.004010	0.947368
4	0.005018	0.003974	0.938053

표준화 적용 후

```
svc = SVC(random_state=777)
scores = cross_val_score(svc, std_x, y, cv=5)
print(scores)
print("교차검증 평균: ", scores.mean())

score_df = DataFrame(cross_validate(svc, std_x, y, cv=5))
score_df
```

```
[0.97368421 0.95614035 1.         0.96491228 0.97345133]
교차검증 평균:  0.9736376339077782
```

	fit_time	score_time	test_score
0	0.003999	0.003000	0.973684
1	0.004997	0.003002	0.956140
2	0.005997	0.002001	1.000000
3	0.005001	0.002998	0.964912
4	0.015003	0.001997	0.973451

최적 파라미터 찾기

```
svc = SVC(random_state=777)
params = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'kernel': ['linear', 'rbf', 'sigmoid', 'poly'],
}

grid_svc = GridSearchCV(svc, param_grid=params, cv=5)
grid_svc.fit(std_x, y)

print(grid_svc.best_params_)

result_df = DataFrame(grid_svc.cv_results_['params'])
result_df['mean_test_score'] = grid_svc.cv_results_['mean_test_score']
result_df.sort_values(by='mean_test_score', ascending=False)
```

```
{'C': 10, 'kernel': 'rbf'}
```

	C	kernel	mean_test_score
17	10.000	rbf	0.977177
8	0.100	linear	0.975408
13	1.000	rbf	0.973638
12	1.000	linear	0.970144
4	0.010	linear	0.968390
16	10.000	linear	0.966651
20	100.000	linear	0.959649
14	1.000	sigmoid	0.959603
19	10.000	poly	0.957864
23	100.000	poly	0.957833
21	100.000	rbf	0.956094
10	0.100	sigmoid	0.949061
9	0.100	rbf	0.945536
0	0.001	linear	0.938534

	C	kernel	mean_test_score
22	100.000	sigmoid	0.931501
18	10.000	sigmoid	0.927977
15	1.000	poly	0.898137
6	0.010	sigmoid	0.894628
11	0.100	poly	0.833085
7	0.010	poly	0.708260
3	0.001	poly	0.643239
1	0.001	rbf	0.627418
5	0.010	rbf	0.627418
2	0.001	sigmoid	0.627418