Voting (2)

지금까지 까지 공부한 분류 알고리즘들 중에서 정확도가 높게 나타나는 상위 2건을 선정하여 Voting 처리 수행

#01. 패키지 참조

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

from matplotlib import pyplot as plt
import seaborn as sb
from pandas import DataFrame, read_excel, melt
from sklearn.ensemble import VotingClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
```

#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	m 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 구분

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

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

데이터 표준화

```
scaler = StandardScaler()
std_x = scaler.fit_transform(x)
std_x.shape
```

```
(569, 30)
```

훈련, 검증 데이터 분리

```
x_train, x_test, y_train, y_test = train_test_split(std_x, y, test_size=0.3, random_st
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

```
((398, 30), (171, 30), (398,), (171,))
```

#04. 높은 정확도를 갖는 상위 2건의 알고리즘 선정하기

알고리즘 실행 함수

```
def singleML(modelName, train_x, train_y, test_x, test_y, cv=5, **kargs):
    # 모델 구축
    model = modelName(**kargs)
    # 학습
    model.fit(train_x, train_y)
    # 훈련 점수
    train_scores = cross_val_score(model, train_x, train_y, cv=cv).mean()
    # 각 훈련 회차별 점수표
    score_df = DataFrame(cross_validate(model, train_x, train_y, cv=5))
    # 검증 데이터에 대한 예측치 생성
    y_pred = model.predict(test_x)
    # 예측치에 대한 정확도 점수
    test_scores = accuracy_score(test_y, y_pred)
    # 리턴
    return [model, train_scores, test_scores, score_df]
```

테스트할 알고리즘 자료구조 정의

```
ml_list = [LogisticRegression, KNeighborsClassifier, DecisionTreeClassifier, SVC]
```

알고리즘별로 테스트

```
scores = []

# 서포트백터머신(SVC)의 경우 독립변수에 이름이 없으면 경고가 표시된다.
# 그래서 이름을 붙여준다. --> 데이터프레임으로 구성
x_train_df = DataFrame(x_train, columns=x.columns)
x_test_df = DataFrame(x_test, columns=x.columns)

for ml in ml_list:
    _, train_score, test_score, _ = singleML(ml, x_train_df, y_train, x_test_df, y_tes
    scores.append({"name": ml.__name__, "train_score": train_score, "test_score": test

df = DataFrame(scores)
df
```

	name	train_score	test_score
0	LogisticRegression	0.987405	0.976608
1	KNeighborsClassifier	0.964747	0.970760
2	DecisionTreeClassifier	0.924557	0.912281
3	SVC	0.977373	0.964912

결과 데이터 프레임 재구조화

```
df2 = melt(df, id_vars=['name'], value_vars=['train_score', 'test_score'], var_name='t
df2
```

	name	type	score
0	LogisticRegression	train_score	0.987405
1	KNeighborsClassifier	train_score	0.964747
2	DecisionTreeClassifier	train_score	0.924557
3	SVC	train_score	0.977373
4	LogisticRegression	test_score	0.976608
5	KNeighborsClassifier	test_score	0.970760
6	DecisionTreeClassifier	test_score	0.912281
7	SVC	test_score	0.964912

알고리즘별 스코어 시각화

```
plt.figure(figsize=(12, 6))
sb.barplot(y='name', x='score', hue='type', data=df2)
plt.legend(bbox_to_anchor=(1, 1))
plt.grid()
```

```
plt.show()
plt.close()
```