Assignment 1: Classification with k-NN

print(X.shape, y.shape)으로 확인 결과, wine dataset의 Input feature의 수는 178개, target class는 13개이다.

KNN 알고리즘에 크게 영향을 주는 hyperparameter로는 distance metric, K의 값, weighting scheme이 있다. Distance metric으로는 Manhattan, Euclidian을 사용했고, weighting scheme으로는 uniform, distance, K의 값은 1부터 10까지 변화시켰다. 그에 따른 결과는 다음과 같았다. Test_train_split을 통해 test_size를 0.25로 설정했고, stratify=y를 통해 y 데이터셋의 target label의 비율을 공평하게 유지했다. 이 과정 없이는 특정 label만이 x나 y 데이터셋이 너무 많이 포함될 수 있고, 그렇다면 어떤 경우에는 test accuracy가 높거나 낮게 나올 수 있다. 따라서 이 수치를 일반화하기는 힘들다. random_state=2021을 통해 랜덤 값을 일정하게 유지했다.

Manhattan / Uniform			Manhattan / Distance		
K	Train accuracy	Test accuracy	K	Train accuracy	Test accuracy
1	1.0	0.8	1	1.0	0.8
2	0.9172932330827067	0.7555555555555555	2	1.0	0.8
3	0.9022556390977443	0.7777777777777778	3	1.0	0.7777777777777778
4	0.8721804511278195	0.7555555555555555	4	1.0	0.8
5	0.8872180451127819	0.8444444444444444	5	1.0	0.8444444444444444
6	0.8345864661654135	0.7777777777777778	6	1.0	0.7777777777777778
7	0.8195488721804511	0.7555555555555555	7	1.0	0.8222222222222222
8	0.8045112781954887	0.7777777777777778	8	1.0	0.8
9	0.8120300751879699	0.7555555555555555	9	1.0	0.7777777777777778
10	0.7969924812030075	0.7333333333333333	10	1.0	0.8222222222222222
Euclidian / Uniform			Euclidian / Distance		
1	1.0	0.7333333333333333	1	1.0	0.7333333333333333
2	0.8721804511278195	0.7333333333333333	2	1.0	0.7333333333333333
3	0.8872180451127819	0.7777777777777778	3	1.0	0.7555555555555555
4	0.8345864661654135	0.6888888888888889	4	1.0	0.7555555555555555
5	0.8045112781954887	0.8	5	1.0	0.8
6	0.7593984962406015	0.7333333333333333	6	1.0	0.7777777777777778
7	0.7593984962406015	0.7111111111111111	7	1.0	0.7777777777777778
8	0.7518796992481203	0.7333333333333333	8	1.0	0.8
9	0.7669172932330827	0.7333333333333333	9	1.0	0.7777777777777778
10	0.7744360902255639	0.7111111111111111	10	1.0	0.8

Uniform 경우에 K가 너무 작은 값일 때는 train accuracy가 높지만 test accuracy가 낮으므로 underfitting됐다는 것을 유추할 수 있고, K가 너무 큰 값일 때는 train accuracy와 test accuracy 모두 낮으므로 overfitting됐다는 것을 유추할 수 있다. Distance를 사용했을 때 train accuracy가 항상 1인 것은 train set에 이미 그 데이터들이 포함돼 있기 때문이다.

Manhattan, Euclidian distance metric 모두 weight scheme으로 uniform을 사용하나 distance를 사용하나 K가 5인 경우에 test accuracy가 가장 높은 것을 알 수 있다. Manhattan의 경우에는 0.84444... Euclidian의 경우에는 0.8을 기록했다. 따라서, weight scheme에는 큰 영향을 받지 않는 dataset임을 알 수 있고, Manhattan과 Euclidian 중에는 Manhattan의 test accuracy가 0.04444... 높으므로, Manhattan distance metric이 더 정확한 결과를 주는 것을 알 수 있었다.

```
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

X, y = load_wine(return_X_y=True)
print(X.shape, y.shape)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, stratify=y,
random_state=2021)
for k in range(1, 11):
    clf = KNeighborsClassifier(p=1, n_neighbors=k, weights='uniform')
    clf.fit(X_train, y_train)

    y_train_hat = clf.predict(X_train)
    y_test_hat = clf.predict(X_test)
    print('M/u k={} train : '.format(k), accuracy_score(y_train, y_train_hat), '/
test : ', accuracy_score(y_test, y_test_hat))
for k in range(1, 11):
    clf = KNeighborsClassifier(p=2, n_neighbors=k, weights='uniform')
    clf.fit(X_train, y_train)

    y_train_hat = clf.predict(X_train)
    y_test_hat = clf.predict(X_test)
    print('E/u k={} train : '.format(k), accuracy_score(y_train, y_train_hat), '/
test : ', accuracy_score(y_test, y_test_hat))
for k in range(1, 11):
    clf = KNeighborsClassifier(p=1, n_neighbors=k, weights='distance')
    clf.fit(X_train, y_train)

    y_train_hat = clf.predict(X_train)
    y_test_hat = clf.predict(X_test)
    print('M/d k={} train : '.format(k), accuracy_score(y_train, y_train_hat), '/
test : ', accuracy_score(y_test, y_test_hat))
for k in range(1, 11):
    clf = KNeighborsClassifier(p=2, n_neighbors=k, weights='distance')
    clf.fit(X_train, y_train)

    y_train_hat = clf.predict(X_train)
    y_test_hat = clf.predict(X_test)
    print('E/d k={} train : '.format(k), accuracy_score(y_train, y_train_hat), '/
test : ', accuracy_score(y_test, y_test_hat))
```