[Lab 6]

실습과제

- **문제.** 이번 실습에서는 k-means 클러스터링 알고리즘을 수행합니다.

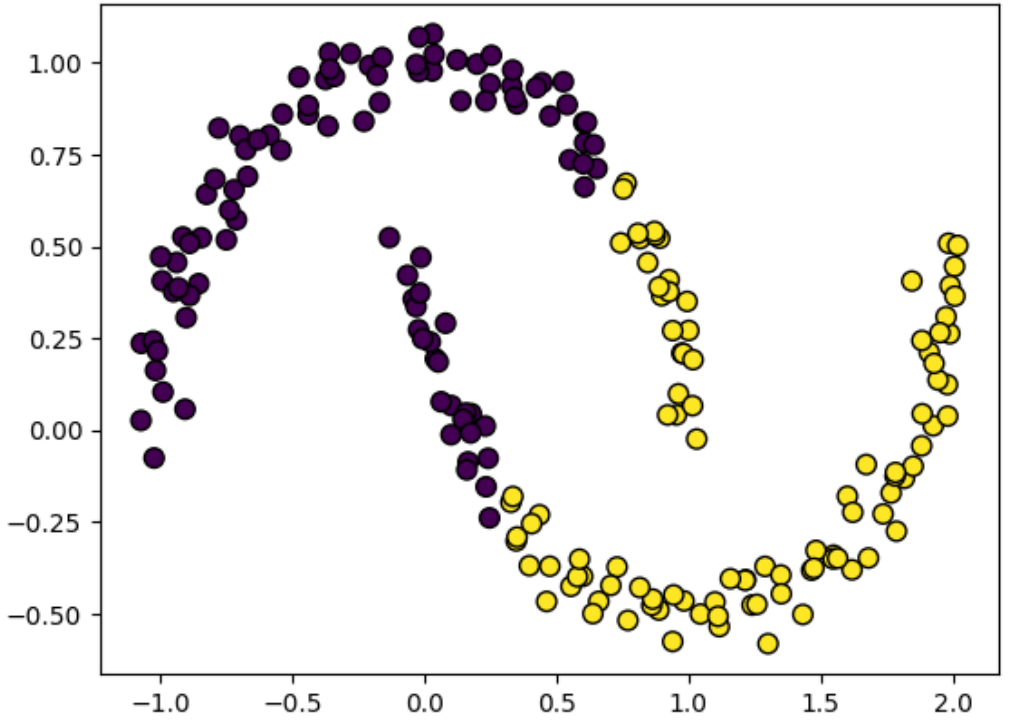
1. 클러스터 수(n_cluster)를 2에서 10까지 바꾸어가면서 실행해보고 plot(Scatter) 그림을 보고서에 첨부합니다.

```
from sklearn.datasets import make_moons
X, y = make_moons(n_samples=200, noise=0.05, random_state=0)
plt.scatter(X[:, 0], X[:, 1], c=y_pred, s=60, edgecolors='k')
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)
y_pred = kmeans.predict(X)
```

실습과제

1. 클러스터 수(n_cluster)를 2에서 10까지 바꾸어가면서 실행해보고 plot 그림을 보고서에 첨부합니다.

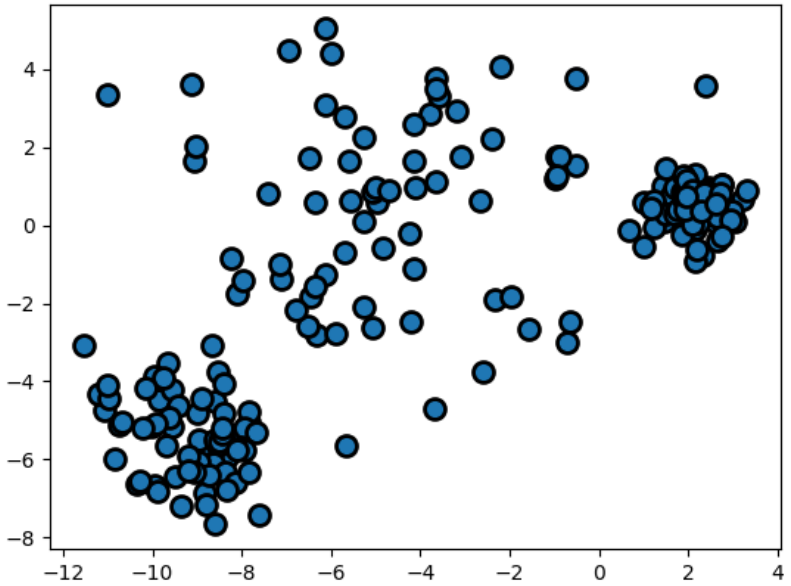
```
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
            marker='^', s=100, linewidth=2, edgecolors='k')
plt.xlabel("특성 0")
plt.ylabel("특성 1")
```



실습과제

2. cluster_std와 n_cluster의 값을 다양하게 변화시켜 가면서 수행해 보고 1번의 plot 코드를 활용하여 그림을 그려보고 보고서에 첨부합니다.

```
X_varied, y_varied = make_blobs(n_samples =200, cluster_std=[1.0, 2.5, 0.5], random_stae=170)
y_pred = KMeans(n_cluster=3, random_state=0).fit_predict(X_varied)
```



실습과제

3. 위의 실험을 수행해 보고 1번과 2번에서 각각 유추해볼 수 있는 k-means clustering algorithm의 단점을 보고서에 서술하세요.

K-Means

- 군집화 : 훈련 집합이 주어지면 조건을 만족하는 군집 집합을 찾아내는 작업입니다.

- K–평균(K-Means)은 주어진 데이터를 k개로 묶는 알고리즘입니다.

- 특징
  - 원리가 단순하고 성능이 좋음
  - 직관적으로 이해하기 쉽고 구현이 쉬움.
  - 군집 개수 k를 설정해야 함.

Sklearn Kmeans()

- KMeans()

```
from sklearn.cluster import KMeans

KMenas(n_clusters=8, *, init='k-means++', n_init=10, max_iter=300, tol=0.0001,
precompute_distances='deprecated', verbose=0, random_state=None, copy_x=True, n_jobs='deprecated',
algorithm='auto')
```

- Parameter

- **n_clusters** : k개, 클러스트의 중심 수 (int, optional, default=8)
- **init** : 초기화 방법을 정해 초기 중앙점을 정합니다. *(default = 'k-means++',random,ndarray)*
- **N_init**: 최상의 결과를 얻기 위해 몇 번 초기값을 변경하여 알고리즘을 수행할지 정합니다.(int) *(default = 10)*
- **max_iter** : 최대 반복 횟수입니다.(int) *(default = 300)*

Sklearn Kmeans()

- KMeans()

- Example

```
from sklearn.cluster import KMeans
import numpy as np


X = np.array([[1, 2], [1, 4], [1, 0],[10, 2], [10, 4], [10, 0]])
kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
# kmeans.labels_array([1, 1, 1, 0, 0, 0], dtype=int32)
kmeans.predict([[0, 0], [12, 3]])
# array([1, 0], dtype=int32)
kmeans.cluster_centers_
#array([[10.,  2.],[ 1.,  2.]])


# https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
```

Sklearn Make_blobls()

- Make_blobs()

```
from sklearn.datasets import make_blobs
make_blobs(n_samples = 100 , n_features = 2 , * , centers = None , cluster_std = 1.0 , center_box = (-10.0 , 10.0) , shuffle = True , random_state = None , return_centers = False)
```

- Parameter
- **n_samples** : 표본 데이터의 수(default =100)
- **n_features :** 샘플의 수(default = 2)
- **centers**: 생성할 클러스터의 수 또는 중심 위치
- **cluster_std** : 군집의 표준편차(default = 1.0)
- **center_box** : 클러스트 중심에 대한 바운딩 박스

- Return

- **X** : [n_samples, n_features] 크기의 배열
- **Y:** [n_samples] 크기의 배열
- **Centers :** return_centers=True일 때만 반환, 각 클러스트의 중심들이 반환된다.

Sklearn Make_blobls()

- Make_blobs()

- Example

```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=10, centers=3, n_features=2, random_state=0)
print(X.shape)
# (10, 2)
print(y)
# array([0, 0, 1, 0, 2, 2, 2, 1, 1, 0])
X, y = make_blobs(n_samples=[3, 3, 4], centers=None, n_features=2, random_state=0)
print(X.shape)
# (10, 2)
print(y)
# array([0, 1, 2, 0, 2, 2, 2, 1, 1, 0])
```

```
# https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html
```

Sklearn Make_blobls()

- Make_blobs()

- Example

```
from sklearn.datasets import make_blobs

plt.title("5 Cluter Data")
X, y = make_blobs(n_samples=500, n_features=2, centers=5, random_state=1)
plt.scatter(X[:, 0], X[:, 1], marker='o', c=y, s=100,
            edgecolor="k", linewidth=2)
plt.xlabel("$X_1$")
plt.ylabel("$X_2$")
plt.show()
```

Sklearn Make_blobls()

- Make_blobs()

- Example