2장. 군집분석

2장. 군집분석

1절. 군집 모델

클러스터링

중심 기반 클러스터링

https://commons.wikimedia.org/wiki/File:KMeans-Gaussian-data.svg

연결 기반 클러스터링(DBSCAN)

https://commons.wikimedia.org/wiki/File:DBSCAN-density-data.svg

2장. 군집분석

2절. K-Means 클러스터링

클러스터(cluster) : 독립 변수의 특성이 유사한 데이터의 그룹

클러스터링(clustering) : 주어진 데이터를 여러 개의 클러스터로 구분하는 것

만약 클러스터의 수가 K라면 클러스터링은 모든 데이터에 대해 1~K번 클러스터 중에서 몇 번 클러스터에 속하는지 예측하는 작업

$$J = \sum_{k=1}^{K} \sum_{i \in C_k} d(x_i, \ \mu_k)$$

K-Means 클러스터링

- 가장 단순하고 빠른 클러스터링 알고리즘의 하나
- 목적함수 값이 최소화될 때까지 클러스터의 중심(centroid) μk와 각 데이터가 소속될 클러스터를 반복해서 찾는 것

sklearn.cluster.KMeans

K-Means 클러스터링 알고리즘을 이용해서 모델을 생성하고 학습시킴

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

K-Means 클러스터링의 세부 알고리즘

- 1. 임의의 중심값 μk를 고릅니다.(보통 데이터 샘플 중의 하나를 선택합니다.)
- 2. 중심에서 각 샘플 데이터까지의 거리를 계산합니다.
- 3. 각 데이터 샘플에서 가장 가까운 중심을 선택하여 클러스터 갱신합니다.
- 4. 다시 만들어진 클러스터에 대해 중심을 다시 계산하고 1~4를 반복합니다.

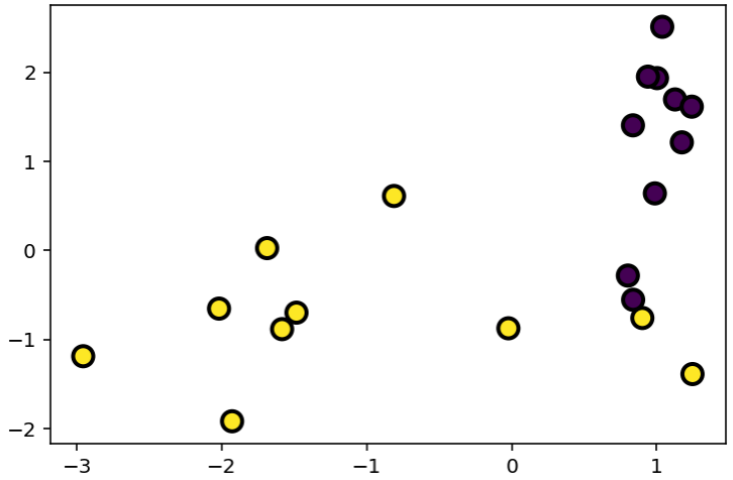
make_classification() 함수는 분류(classification)를 위한 가상 데이터셋을 만드는 함수

```
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
import matplotlib.pyplot as plt
```

```
from sklearn.datasets import make_classification
```

```
X, y = make_classification(n_samples=20, n_features=2, n_informative=2,
                           n_redundant=0, n_clusters_per_class=1,
                           n_classes=2, random_state=123)
```

```
plt.scatter(X[:,0], X[:,1], marker='o', c=y, s=100, edgecolors='k', linewidth=2)
plt.show()
```



클러스터링 모형

모형 생성

```
1  from sklearn.cluster import KMeans
2  model = KMeans(n_clusters=2, init="random")
3  model.fit(X)
```

```
KMeans(algorithm='auto', copy_x=True, init='random', max_iter=300,
    n_clusters=2, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

중심점

```
1  model.cluster_centers_
```

```
array([[ 1.01138251,  0.83200493],
       [-1.56258716, -0.69768199]])
```

군집 예측

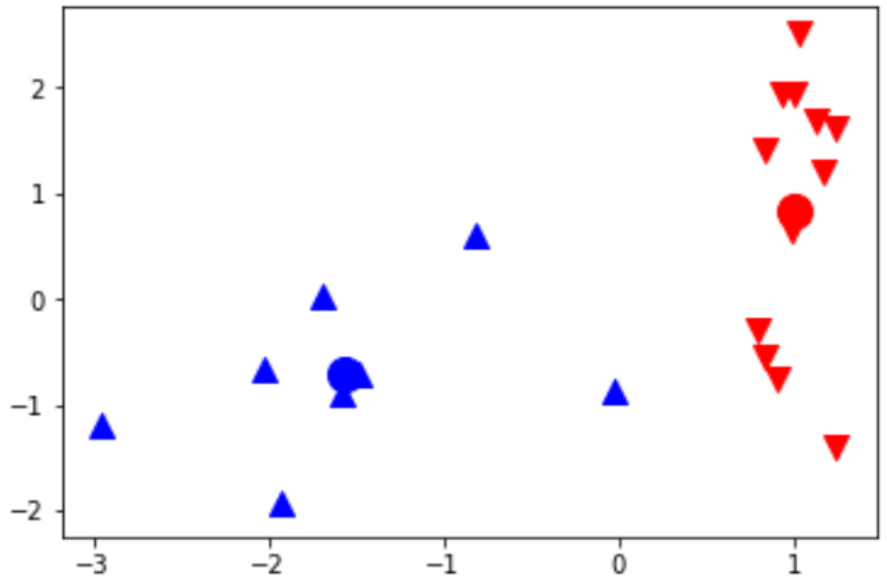
```
1  pred = model.predict(X)
2  pred
```

```
array([0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0])
```

클러스터링 결과 시각화


```
c0, c1 = model.cluster_centers_
plt.scatter(x=X[model.labels_ == 0, 0], y=X[model.labels_ ==0, 1],
            s=100, marker='v', c='r')
plt.scatter(x=X[model.labels_ == 1, 0], y=X[model.labels_ ==1, 1],
            s=100, marker='^', c='b')
plt.scatter(x=c0[0], y=c0[1], s=200, c='r')
plt.scatter(x=c1[0], y=c1[1], s=200, c='b')
plt.show()
```

모델을 이용해 클러스터링 된 결과를 산점도를 이용해 표시



회차별 군집 확인

KMeans 클래스의 max_iter 인자는 최대 학습 횟수를 지정

중심점을 찾아가는 과정을 시각화

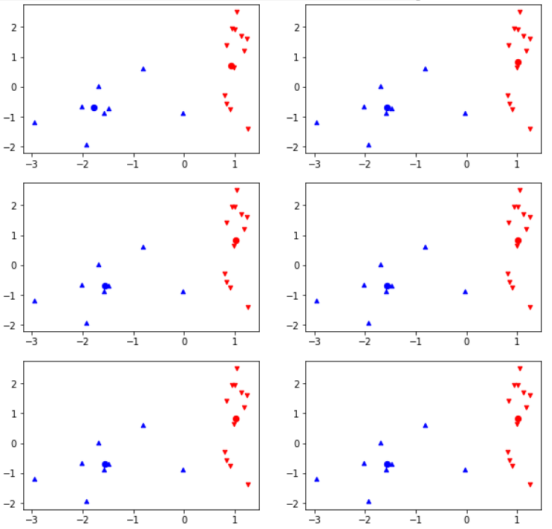
모델과 데이터를 인수로 받아 산점도와 중심점을 출력할 함수

```
def plot_cluster(model, data):
    c0, c1 = model.cluster_centers_
    plt.scatter(X[model.labels_ == 0, 0], X[model.labels_ ==0, 1],
                s=100, marker='v', c='r')
    plt.scatter(X[model.labels_ == 1, 0], X[model.labels_ ==1, 1],
                s=100, marker='^', c='b')
    plt.scatter(c0[0], c0[1], s=200, c='r')
    plt.scatter(c1[0], c1[1], s=200, c='b')
#     plt.show()
```

회차별 군집 확인

1회부터 6회까지 진행하는 모델을 만든 후 각 모델을 이용해 계산한 중심점과 클러스터링 된 결과를 시각화

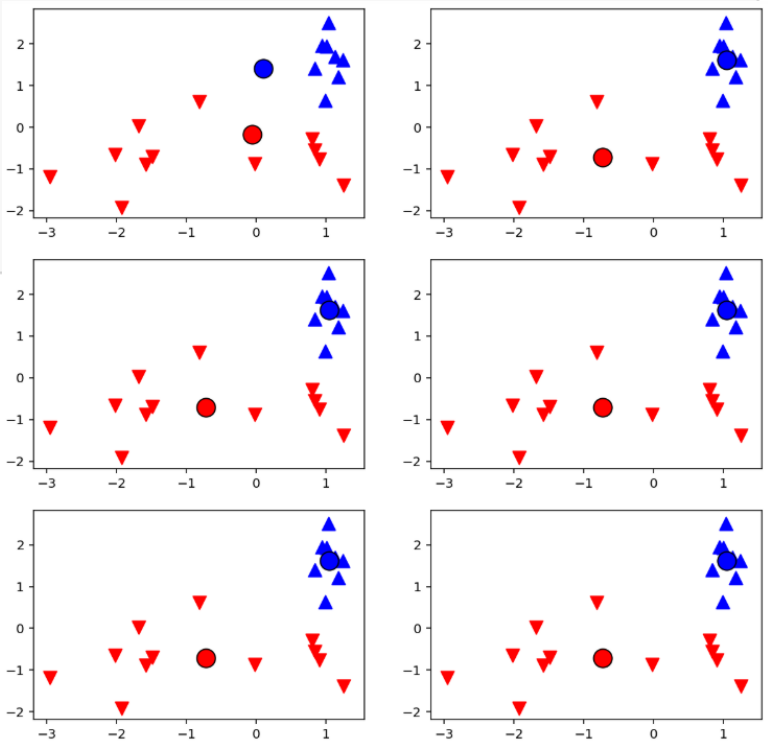
```
plt.figure(figsize=(10,10))
model1 = KMeans(n_clusters=2, init="random", n_init=1,
                max_iter=1, random_state=1)
model1.fit(X)
plt.subplot(3,2,1)
plot_clusters(model1, X)

model2 = KMeans(n_clusters=2, init="random", n_init=1,
                max_iter=2, random_state=1)
model2.fit(X)
plt.subplot(3,2,2)
plot_clusters(model2, X)
```



- 동일한 코드를 작성
- 변수 이름과 숫자만 1부터 6까지 바꿔서 코드 작성 후 실행

반복문을 이용한 회차별 군집 확인


```
plt.figure(figsize=(10,10))
for i in range(6):
    model = KMeans(n_clusters=2, init="random",
                   n_init=1, max_iter=(i+1),
                   random_state=1)
    model.fit(X)
    plt.subplot(3,2,(i+1))
    plot_cluster(model, X)
```



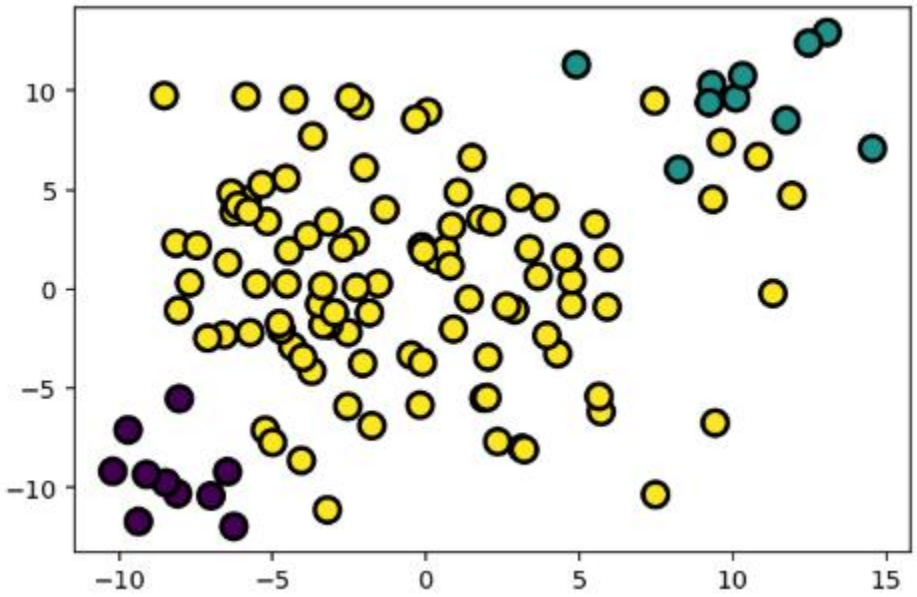
k-Means 클러스터링의 한계점

군의 특성이 다를 경우

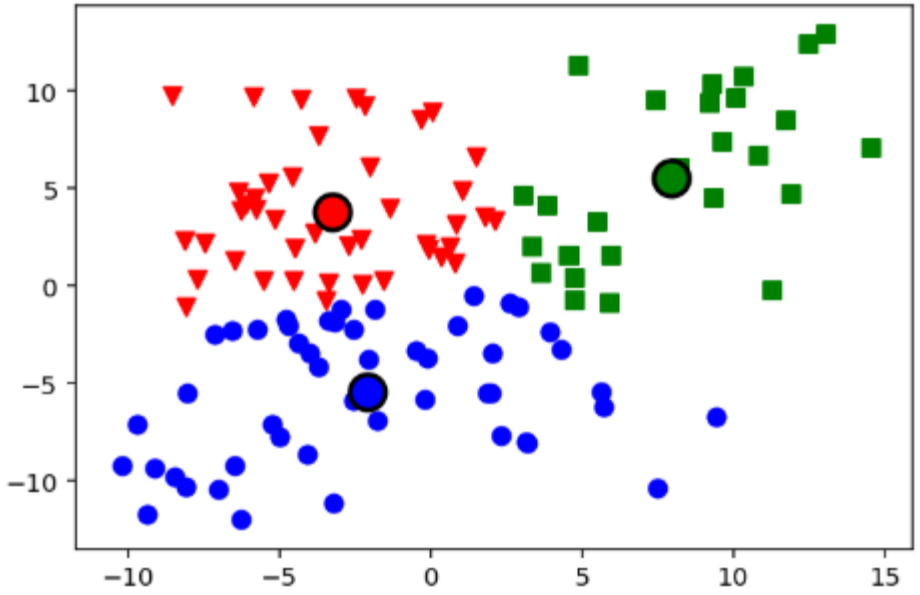
- 크기(Sizes)
- 밀도(Densities)
- 비 구형(Non-globular shapes)

이상치를 포함할 경우

군의 크기가 다를 경우

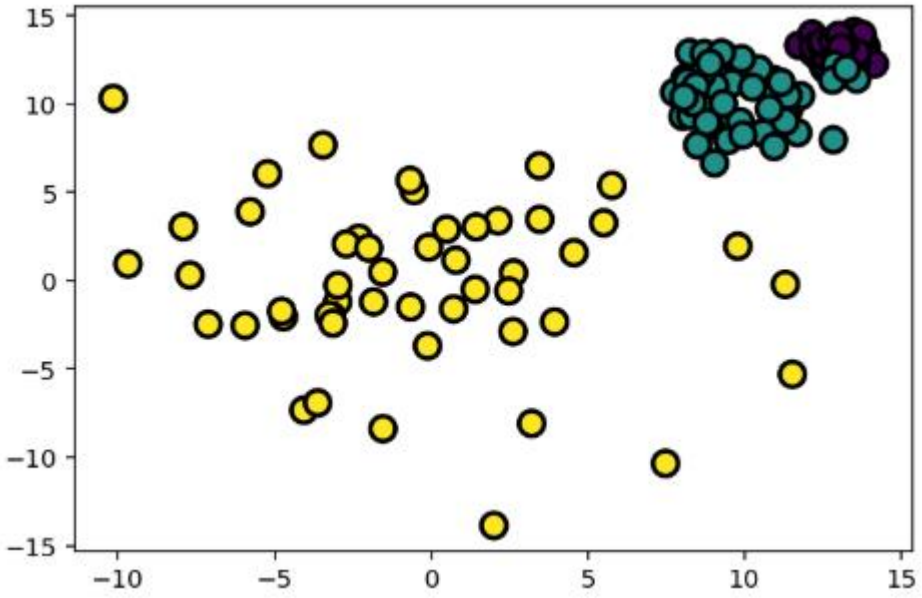


Original Points

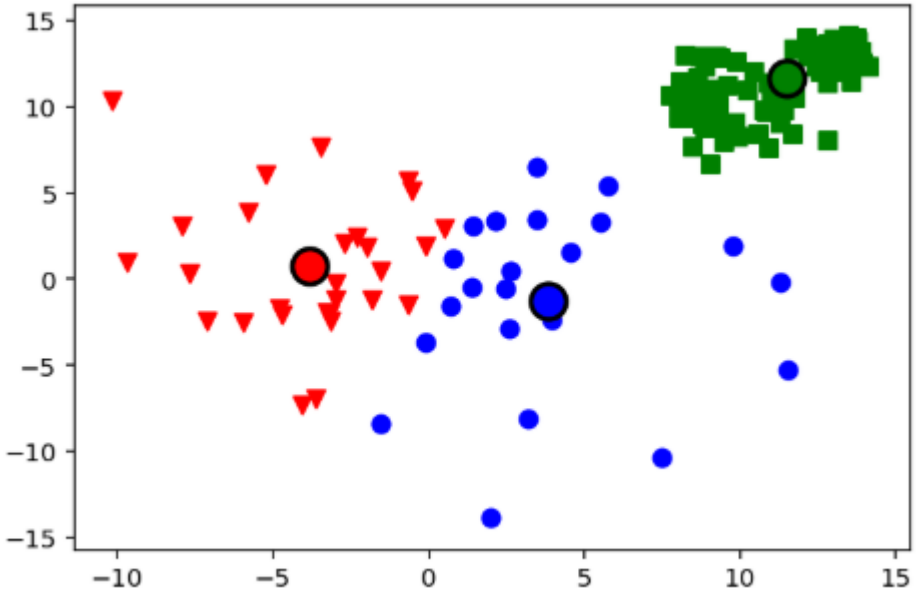


k-means (3 Clusters)

군의 밀도가 다를 경우

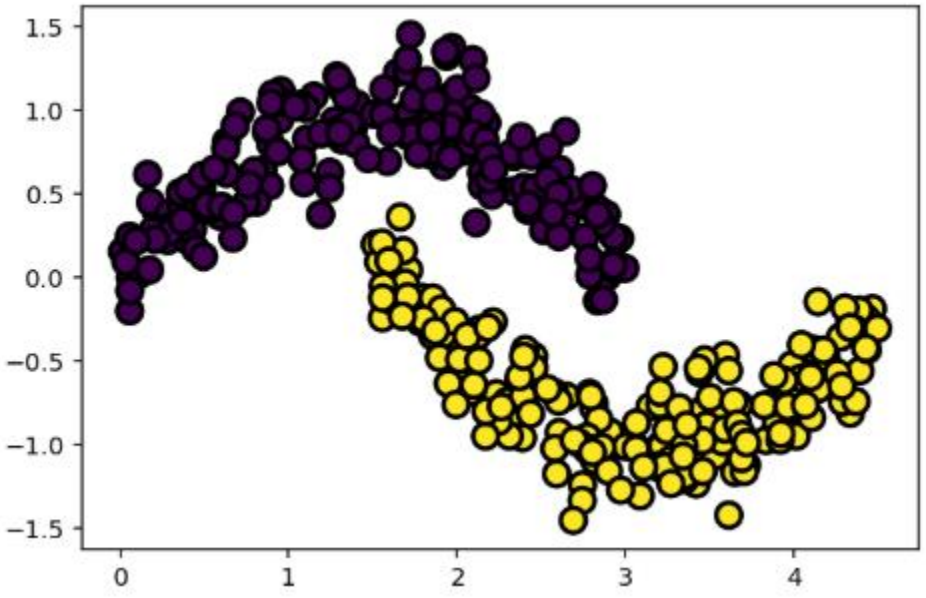


Original Points

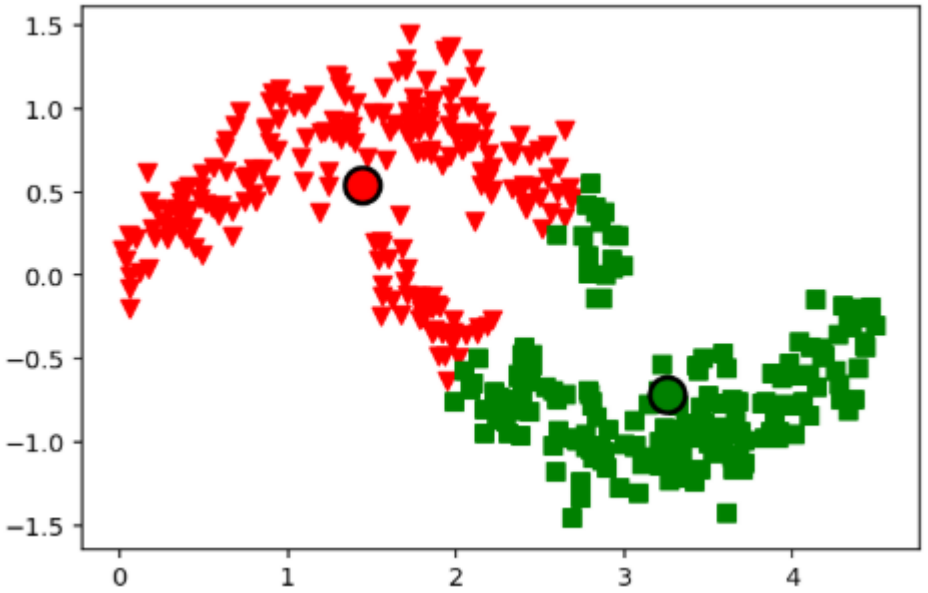


k-means (3 Clusters)

군이 구형이 아닌 경우

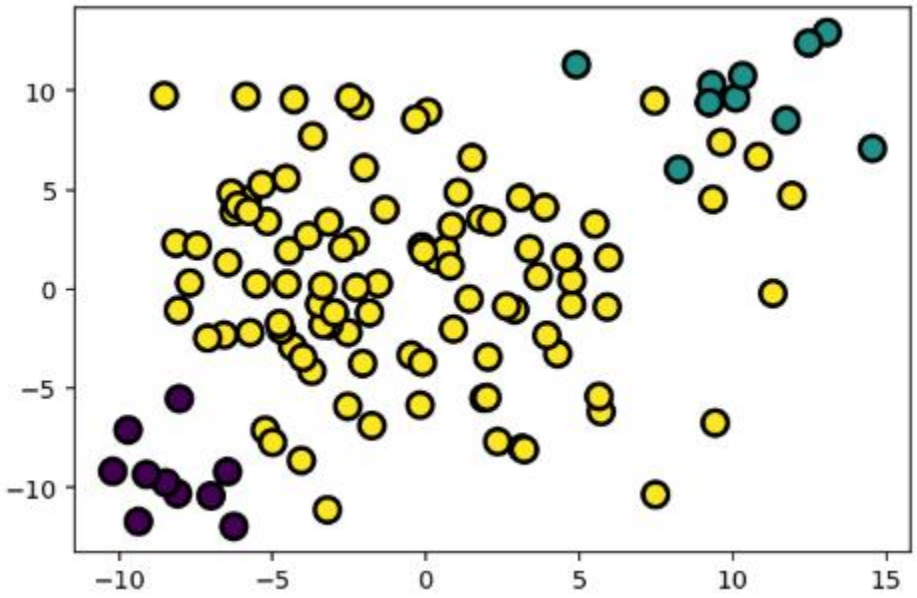


Original Points

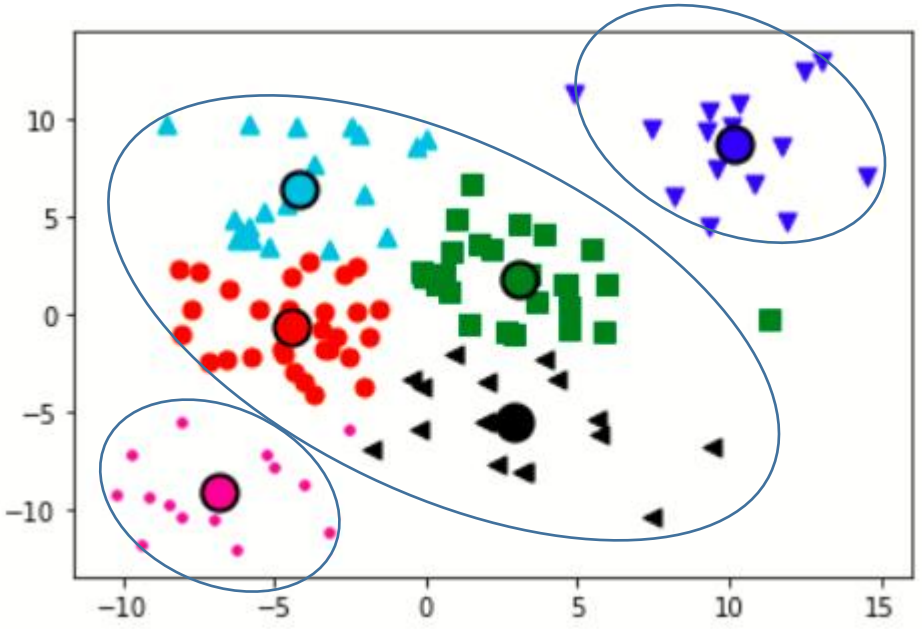


k-means (2 Clusters)

군의 크기가 다를 경우 해결

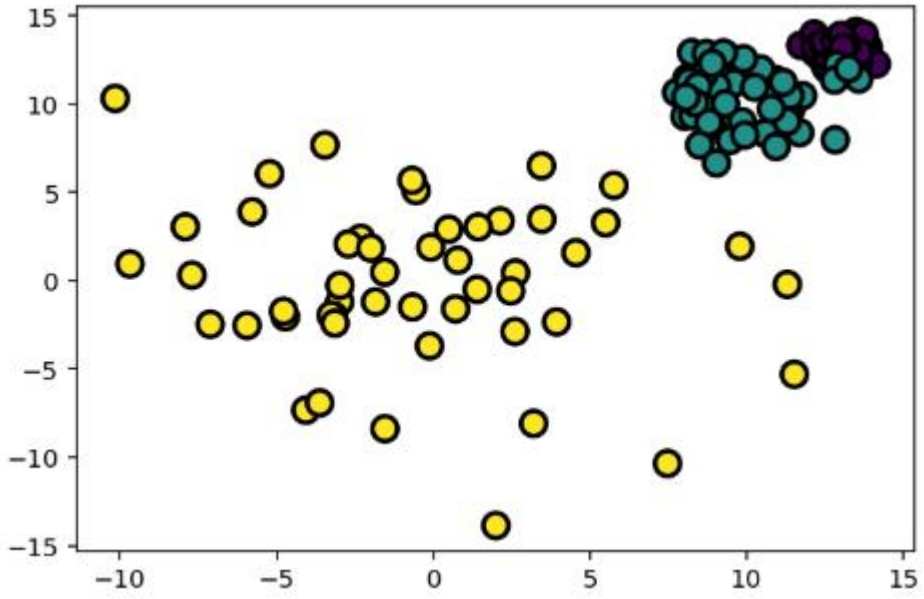


Original Points

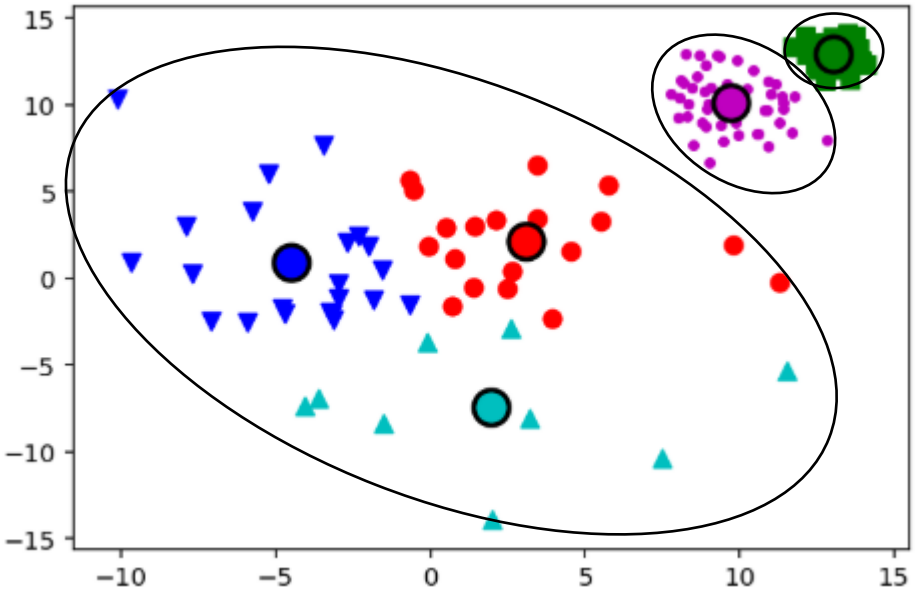


k-means (6 Clusters)

군의 밀도가 다를 경우 해결

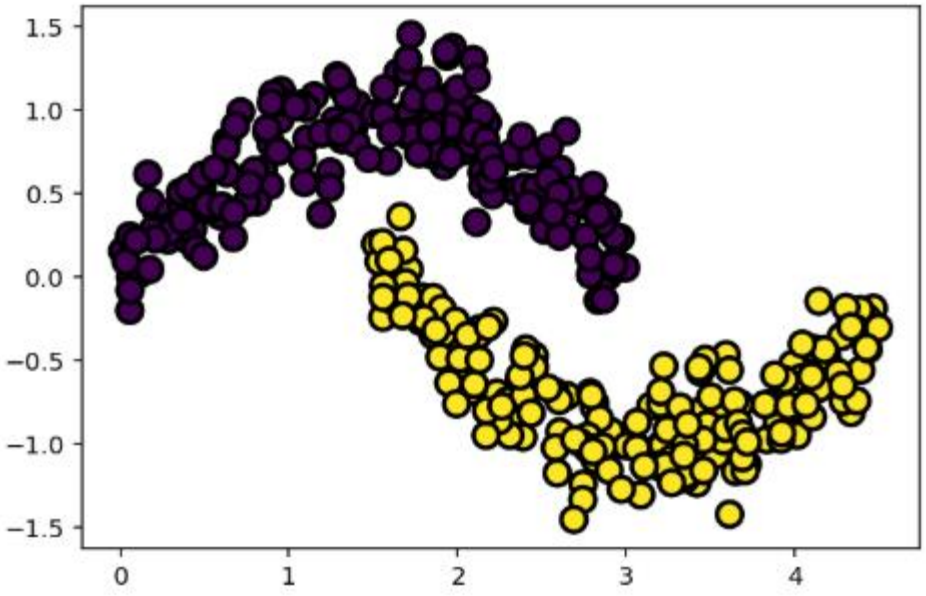


Original Points

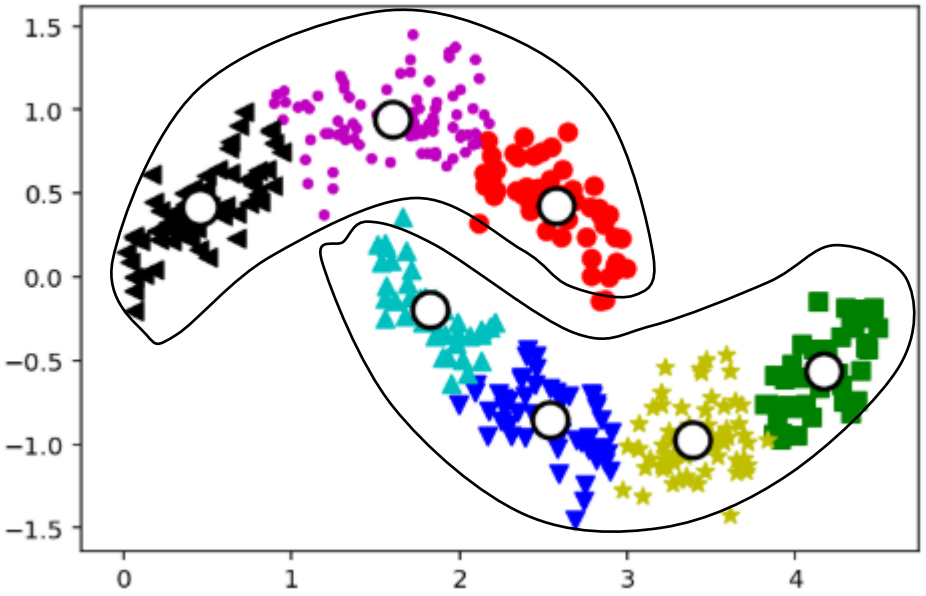


k-means (5 Clusters)

군이 구형이 아닌 경우 해결



Original Points



k-means (7 Clusters)

2장. 군집분석

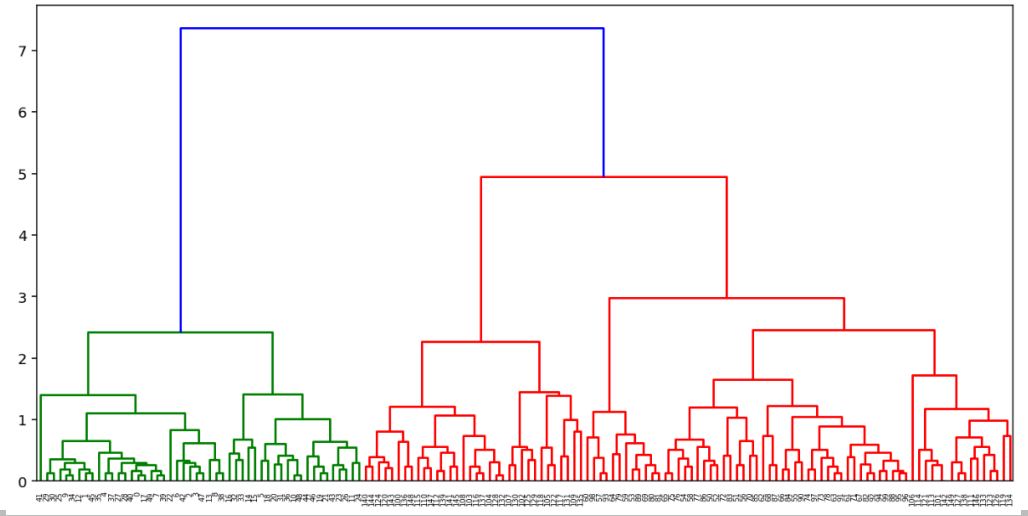
3절. Hierarchical 클러스터링

계층적 군집

3절. Hierarchical 클러스터링

- 비슷한 군끼리 묶으면서 하나의 군집이 될 때까지 군을 묶는 알고리즘
- 군집간의 거리를 기반으로 클러스터링을 하는 알고리즘
- 군집의 수를 미리 정해주지 않아도 됨
- dendrogram이라는 그래프를 이용하면 손쉽게 시각화

Dendrogram



계층적 군집

3절. Hierarchical 클러스터링

```
1 import seaborn as sns
2 iris = sns.load_dataset("iris")
```

데이터 불러오기

```
1 from sklearn.preprocessing import LabelEncoder
2 le = LabelEncoder()
3 le.fit(iris.species)
4 iris["species"] = le.transform(iris.species)
```

레이블 인코딩

```
1 from scipy.cluster.hierarchy import linkage
2 cluster_model = linkage(iris, method="complete")
```

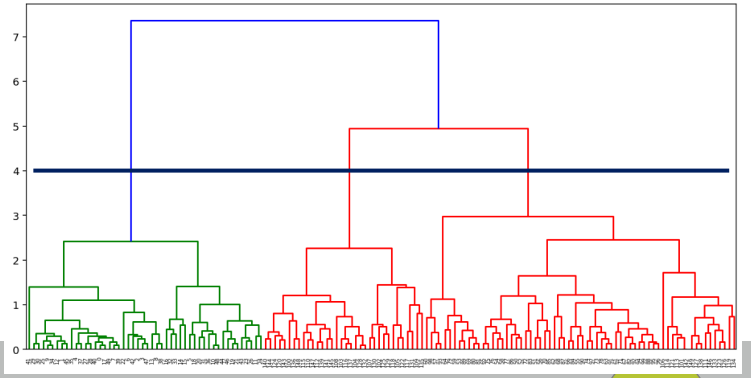
계층적 군집 실시

```
1 from scipy.cluster.hierarchy import dendrogram
2 plt.figure(figsize=(12,6))
3 dendrogram(cluster_model, labels=iris.index)
4 plt.show()
```

덴드로그램으로 표현

계층 분석을 통한 군집의 수 결정

- 계층적 군집은 최종적으로 1개의 군집으로 모든 데이터를 클러스터링
- 실제로는 n개의 군집으로 나눠야 함
- dendrogram에서 보면 위로 올라갈 수 록 클러스터는 병합
- 적정한 y값에서 클러스터링을 멈추면 n개의 군까지만 클러스터링이 됨
- y=4(수평선이 표시된 곳)에서 클러스터링이 멈추면 총 3개의 클러스터로 군집이 됨



fcluster() 함수

fcluster() 함수를 이용해서 y값을 지정해서 클러스터링을 멈추게 함

```
1 from scipy.cluster.hierarchy import fcluster
2 fcluster(cluster_model, 4, criterion="distance")
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 2, 2, 2, 2, 3, 2, 2, 2,
       2, 3, 2, 3, 3, 2, 2, 2, 2, 3, 2, 3, 2, 3, 2, 2, 3, 3, 2, 2, 2, 2,
       2, 3, 3, 2, 2, 2, 3, 2, 2, 2, 3, 2, 2, 2, 3, 2, 2, 3], dtype=int32)
```

```
1  fcluster(cluster_model, 2, criterion="distance")
```

```
array([1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 2, 1, 1, 2, 2, 2, 1, 2, 2, 2, 2,
       1, 2, 2, 1, 2, 1, 1, 1, 1, 2, 2, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2,
       2, 1, 2, 1, 2, 1, 6, 6, 6, 5, 6, 6, 6, 5, 6, 5, 5, 6, 6, 6, 5, 6,
       6, 6, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 5, 5, 6, 6, 6, 6, 6, 6,
       6, 5, 6, 6, 6, 5, 6, 6, 6, 6, 5, 6, 3, 7, 4, 3, 3, 4, 7, 4, 3, 4,
       3, 7, 3, 7, 7, 3, 3, 4, 4, 7, 3, 7, 4, 7, 3, 4, 7, 7, 3, 4, 4, 4,
       3, 7, 7, 4, 3, 3, 7, 3, 3, 3, 7, 3, 3, 3, 7, 3, 3, 7], dtype=int32)
```

y값을 낮게 하면 더 많은 클러스터가 만들어짐

예측한 데이터를 이용한 시각화


```
from scipy.cluster.hierarchy import fcluster
predict = fcluster(cluster_model, 4, criterion='distance')
```

y=4로 군집 예측

```
import numpy as np
adjusted_pred = np.choose((predict-1), [0, 2, 1])
adjusted_pred
```

클러스터링 할당 번호 재조정

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2,
       2, 1, 2, 1, 1, 2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 2, 1, 1, 2, 2, 2, 2,
       2, 1, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 1])
```

```
import pandas as pd
pred_name = le.inverse_transform(adjusted_pred)
origin_name = le.inverse_transform(iris.species.values)
pd.crosstab(origin_name, pred_name,
            rownames=["True"],
            colnames=["Predicted"],
            margins=True)
```

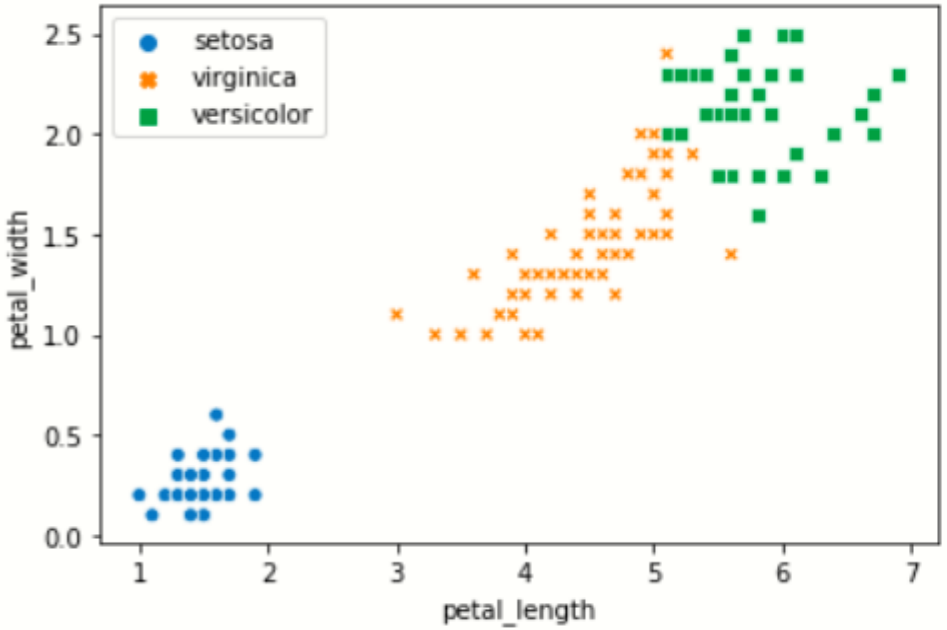
교차분류표 출력

Predicted True	setosa	versicolor	virginica	All
setosa	50	0	0	50
versicolor	0	50	0	50
virginica	0	16	34	50
All	50	66	34	150

예측한 데이터를 이용한 시각화


```
1 import seaborn as sns
```

```
1 ax = sns.scatterplot(x="petal_length", y="petal_width",
2                       hue=le.inverse_transform(predict-1),
3                       style=le.inverse_transform(predict-1),
4                       data=iris)
```



산점도 출력

2장. 군집분석

4절. DBSCAN 클러스터링

2장. 군집분석

5절. 군집모형 성능평가

Scikit-learn의 모형 평가 방법

예측 모형의 score 메소드 (분류모형에서)

- 예측 모형들은 score() 메소드를 통해 예측 모형을 평가할 수 있는 기본 기준을 제공합니다.

metrics 함수

- 메트릭(sklearn.metrics) 모듈은 분류, 회귀 그리고 군집모형 등 예측 모형의 평가를 위한 함수들을 제공합니다.

scoring 매개변수

- sklearn.model_selection 모듈의 cross_val_score() 함수 또는 GridSearchCV 클래스 등 교차 검증(cross-validation)을 사용하는 모형 평가 도구들은 내부적으로 scoring 매개변수를 이용해 모형 평가 규칙을 정의합니다.
- scoring 매개변수의 가능한 값은 sklearn.metrics.SCORES.keys() 함수를 통해 알 수 있습니다.

클러스터링 성능평가 기준

KMeans 클러스터링의 score() 함수는 inertia_ 속성의 값을 음수로 출력하기 때문에 클러스터링의 모델 평가 수치를 확인하려면 다른 방법을 사용해야 함

scoring 속성의 값	군집 모형 평가 함수	참고
'adjusted_mutual_info_score'	metrics.adjusted_mutual_info_score	
'adjusted_rand_score'	metrics.adjusted_rand_score	
'completeness_score'	metrics.completeness_score	
'fowlkes_mallows_score'	metrics.fowlkes_mallows_score	
'homogeneity_score'	metrics.homogeneity_score	
'mutual_info_score'	metrics.mutual_info_score	
'normalized_mutual_info_score'	metrics.normalized_mutual_info_score	
'v_measure_score'	metrics.v_measure_score	

클러스터링 성능평가 기준

- Adjusted Rand index :
  - Rand Index(랜드 지수)는 주어진 N개의 데이터 중에서 2개를 선택해 이 쌍(pair)이 클러스터링 결과 U와 V에서 모두 같은 클러스터에 속하는지, 서로 다른 클러스터에 속하는지를 확인
  - Rand Index는 클러스터 수가 많아지면 b값이 커질 확률이 크고, rand index도 높은 값을 가짐
  - 그래서 Rand Index를 확률적으로 조정한 Adjusted Rand Index를 사용함

$$\overbrace{ARI}^{Adjusted\ Index} = \frac{\overbrace{\sum_{ij}\binom{n_{ij}}{2}}^{Index} - \overbrace{\left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right]/\binom{n}{2}}^{Expected\ Index}}{\underbrace{\frac{1}{2}\left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right]}_{Max\ Index} - \underbrace{\left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right]/\binom{n}{2}}_{Exptected\ Index}}$$

```
1 from sklearn.metrics import adjusted_rand_score
```

```
1 adjusted_rand_score(y_true, iris_cluster_model.labels_)
```

```
0.920405050901892
```

클러스터링 성능평가 기준

- Mutual Information(상호 정보)
  - 정보학이나 확률론에서 두 확률 변수간의 상호 의존도를 나타내는 지표
  - 확률변수 X와 Y가 존재할 때, X를 통해 Y에 대한 정보를 얼마나 얻을 수 있는가를 의미하는 것
  - 결합확률분포 P(X, Y)와 각 변수의 marginal distribution의 곱 P(X)*P(Y)이 얼마나 유사한가로 측정됨
- Normalized Mutual Information(정규화된 MI)
  - Mutual Information 값이 0과 1의 사이 값이 되도록 upper bound 값을 기준으로 정규화한 지표
  - 이때 upper bound는 U와 V가 가진 엔트로피(불확실성)의 산술평균값 혹은 기하평균, 최대/최솟값 등을 사용할 수 있음
- Adjusted Mutual Information
  - Normalized Mutual information이 0과 1사이의 값을 갖더라도 여전히 클러스터 수가 증가하면 실제 상호의존도와 상관없이 값이 증가하는 경향이 있음
  - 따라서 최근에는 상호의존도의 기대값을 이용해 각 클러스터에 할당될 확률값(chance)으로 조정한 Adjusted Mutual information을 주로 사용
  - AMI는 두 클러스터링 결과가 랜덤한 경우 0에 가깝고, 할당 결과가 동일한 경우 1이 되도록 합니다.

```
1 from sklearn.metrics import mutual_info_score
```

```
1 mutual_info_score(iris.species, predict)
```

```
0.8255910976103356
```

$$X = \{X_1, X_2, \ldots, X_r\}$$

$$Y = \{V_1, V_2, \ldots, V_s\}$$

$$P(i) = \frac{|X_i|}{N}$$

$$P(i,j) = \frac{|X_i \cap Y_j|}{N}$$

$$MI(X, Y) = \sum_{i=1}^{r}\sum_{j=1}^{s} P(i,j)\log\frac{P(i,j)}{P(i)P'(j)}$$

클러스터링 성능평가 기준

- homogeneity, completeness, V-measure
  - homogeneity: 각 클러스터가 단일 클래스의 데이터만 가지는 정도
  - completeness: 같은 클래스의 값이 하나의 클러스터로 모여 있는 정도
  - V-measure: homogeneity와 completeness의 조화 평균

$$h = 1 - \frac{H[C|K]}{H[C]}$$

$$c = 1 - \frac{H[K|C]}{H[K]}$$

$$v = 2 \cdot \frac{h \cdot c}{h + c}$$

```
1 from sklearn.metrics import homogeneity_score
2 homogeneity_score(iris.species, predict)
```

0.7514854021988338

```
1 from sklearn.metrics import completeness_score
2 completeness_score(iris.species, predict)
```

0.7649861514489815

```
1 from sklearn.metrics import v_measure_score
2 v_measure_score(iris.species, predict)
```

0.7581756800057784

클러스터의 개수 및 소속을 모르는 경우 실루엣 계수(Silhouette Coefficient)를 사용

$$s = \frac{b - a}{\max(a, b)}$$

a : 같은 클러스터에 속한 원소들의 평균 거리
b : 다른 클러스터 중 가장 가까운 클러스터까지의 평균 거리

silhouette_score() 함수는 모든 샘플의 평균 실루엣 계수를 계산

```
sklearn.metrics.silhouette_score(X, labels,
        metric='euclidean', sample_size=None,
        random_state=None, **kwds)
```

silhouette_sample() 함수는 각 샘플에 대한 실루엣 계수를 계산

```
sklearn.metrics.silhouette_samples(X, labels,
        metric='euclidean', **kwds)
```

1. iris 데이터의 petal_length열과 petal_width열을 이용해서 K-Means 알고리즘으로 군집분석하고 그래프로 시각화하세요(단, 각 클러스터의 중심점이 함께 표시되고 군의 수는 2로 설정)