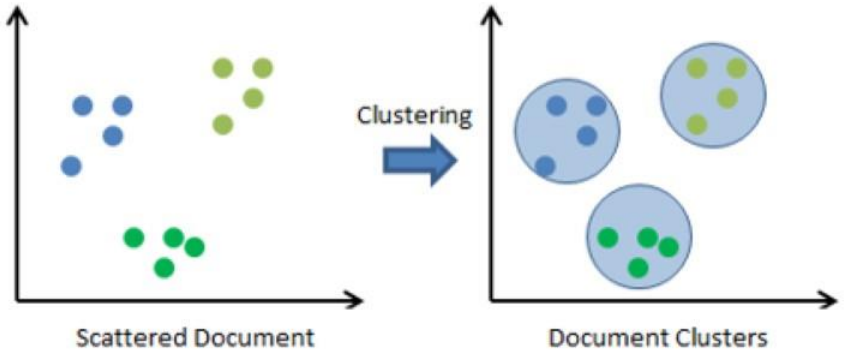
II. 머신러닝 모델

□ Clustering(군집분석)이란

- 각 데이터의 유사성을 측정하여 유사성이 높은 대상 집단을 분류하고, 군집 간에 상이성을 규명하는 방법
  - 군집 간 분산(inter-cluster variance) 최대화
  - 군집 내 분산(inner-cluster variance) 최소화



- 활용분야
  - 페이스북, 구글, 링크드인 등 고객 segmentation을 통한 마케팅 활용

□ Clustering(군집분석)

- 종류
  - 분할기반 군집 모델
    - K-means clustering : 데이터를 사용자가 지정한 k개의 군집으로 나눔
    - DBSCAN : k개를 설정할 필요 없이 군집화 할 수 있는 방법
  - 계층적 군집 모델
  - Hierarchical clustering : 나무 모양의 계층 구조를 형성해나가는 방법

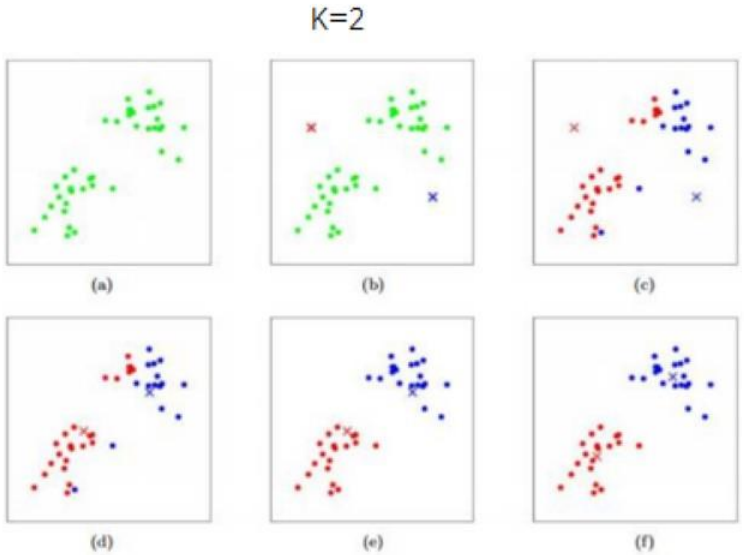
- 주의점
  - 군집분석 모델은 섞여있는 데이터를 분류만 해줌
  - 분류가 지도학습의 분류모델(classification)과 같은 의미는 아님

☐ K-means clustering

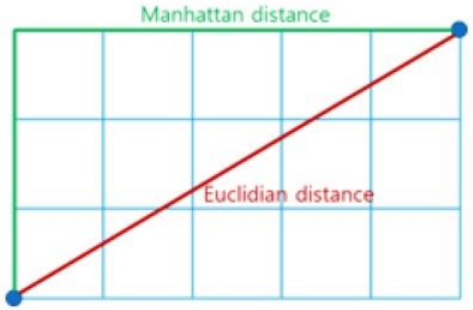
- Label 없이 데이터의 군집으로 k개를 생성

K-means 알고리즘

1. 클러스터 개수 결정(k=n) 후 임의의 중심 n개 설정
2. 모든 데이터는 n개의 중심까지 각각 거리를 계산한 후 가장 가까운 중심을 자신의 클러스터 중심이라고 정함
3. 각 클러스터마다 학습 데이터의 좌표값 평균을 계산한 후 이를 새로운 중심으로 설정
4. 새로 보정후 이동된 중심을 기준으로 2, 3 단계 반복
5. 모든 학습 데이터중에서 자신이 속하는 클러스터를 변경하는 경우가 발생되지 않으면 학습 완료



- 점과 점 사이의 거리 측정
  - Euclidian distance
  - Manhattan distance



☐ K-means clustering

- K 값을 설정하는 방법
  - 군집의 개수 k는 사용자가 임의로 정하는 것이기 때문에 데이터에 최적화된 k를 찾기 어려움
  - K를 설정하는 대표적인 방법은 Elbow method, Silhouette method등이 있음

- Elbow method
  - 군집간 분산과 전체 분산의 비율

$$ratio = \frac{BSS}{TSS} = \frac{TSS - WSS}{TSS}$$

TSS (Total Sum of Squares)

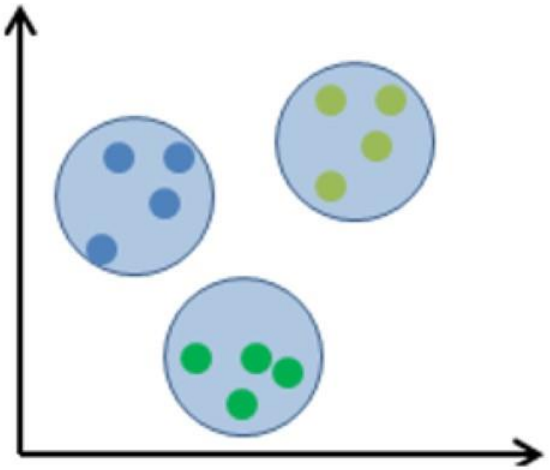
$$= \sum_{i=1}^{N} d(x_i, c)^2$$

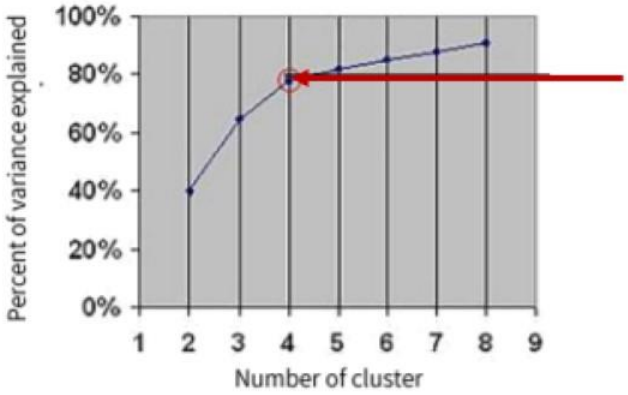
객체 x_i와 전체 데이터의 중심c와의 거리 제곱합

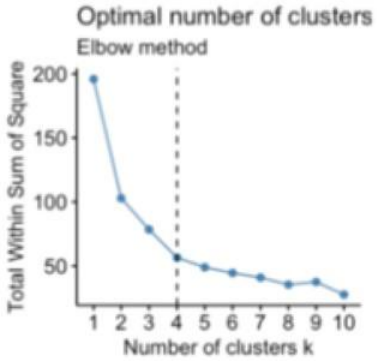
WSS (Within cluster Sum of Squares)

$$= \sum_{j=1}^{K} \sum_{i \in c_j} d(x_i, c_j)^2$$

객체 x_i와 군집 j의 중심 c_j와의 거리 제곱합







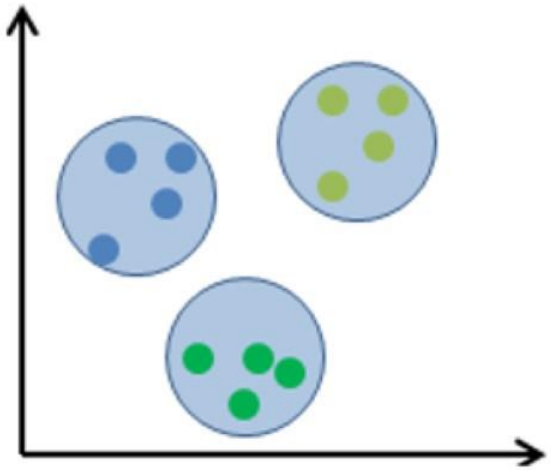
☐ K-means clustering

- K 값을 설정하는 방법
  - Silhouette method
    - 객체와 그 객체가 속한 군집의 데이터들과의 거리를 계산하는 방법

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \qquad \text{where } -1 \le s(i) \le 1$$

$$a(i) = \frac{1}{|g(x_i)| - 1} \sum_{j \in g(x_i)} d(x_i, x_j)$$

$$b(i) = \min_k \left(\frac{1}{|g_k|} \sum_{j \in g_k} d(x_i, x_j) \right)$$



☐ K-means clustering

- 단점
  - 클러스터의 개수를 정해주어야 함
  - 초기 중간값에 민감함
  - 오목한 분포를 갖는 데이터에는 잘 맞지 않는 경향
  - 동떨어져 있는 데이터(아웃라이어)나 노이즈에 민감하게 반응

- K-medoids
  - K-means의 단점을 개선하기 위한 방법
  - 클러스터의 중심을 좌표평면상 임의의 점이 아니라 데이터 셋트의 값 중 하나로 선택
  - 아웃라이어와 노이즈 처리에 좋음
  - 단점
    - 단순한 좌표값의 평균으로 중심을 찾는 k-means에 비해 계산량이 많음
    - 오목한 분포를 갖는 데이터에는 잘 맞지 않음
    - 초기에 클러스터의 수를 정해주어야 함

□ [실습] iris 군집화

- Scikit-learn 에 내장되어 있는 데이터셋 iris 이용
  - 데이터준비

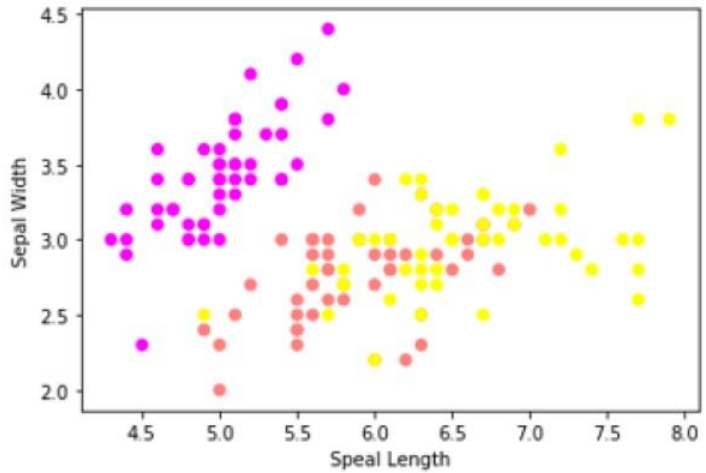
```
from sklearn import datasets
```

```
iris = datasets.load_iris()
```

```
x_data = iris.data[:,:2] #꽃받침(sepal)의 길이와 너비
y_data = iris.target
```

```
plt.scatter(x_data[:,0], x_data[:,1], c = y_data, cmap = "spring")
plt.xlabel('Speal Length')
plt.ylabel('Sepal Width')
```

```
Text(0, 0.5, 'Sepal Width')
```



□ [실습] iris 군집화

- Scikit-learn 에 내장되어 있는 데이터셋 iris 이용
  - 모델 생성

```
from sklearn.cluster import KMeans
```

```
km = KMeans(n_clusters = 3, random_state=102)
km.fit(x_data)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=102, tol=0.0001, verbose=0)
```

  - 중심점 확인

```
centers = km.cluster_centers_
print(centers)
```

```
[[5.006      3.428     ]
 [6.81276596 3.07446809]
 [5.77358491 2.69245283]]
```

□ [실습] iris 군집화

- Scikit-learn 에 내장되어 있는 데이터셋 iris 이용
  - 군집결과 확인

```
new_labels = km.labels_
new_labels
```

```
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, 0, 0, 1, 1, 1, 2, 1, 2, 1, 2, 1, 2, 2, 2, 2, 2, 2, 1,
       2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1,
       1, 1, 1, 2, 2, 1, 1, 1, 1, 2, 1, 2, 1, 2, 1, 1, 2, 2, 1, 1, 1, 1,
       1, 2, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 2])
```

□ [실습] iris 군집화

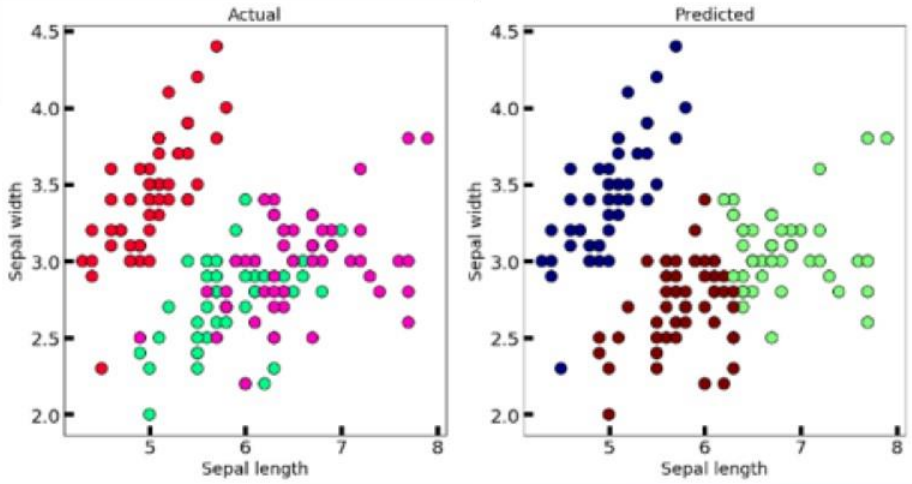
- Scikit-learn 에 내장되어 있는 데이터셋 iris 이용
  - 군집결과 확인

```
fig, axes = plt.subplots(1, 2, figsize=(16,8))
axes[0].scatter(x_data[:, 0], x_data[:, 1], c=y_data, cmap='gist_rainbow',
                edgecolor='k', s=150)
axes[1].scatter(x_data[:, 0], x_data[:, 1], c=new_labels, cmap='jet',
                edgecolor='k', s=150)

axes[0].set_xlabel('Sepal length', fontsize=18)
axes[0].set_ylabel('Sepal width', fontsize=18)
axes[1].set_xlabel('Sepal length', fontsize=18)
axes[1].set_ylabel('Sepal width', fontsize=18)

axes[0].tick_params(direction='in', length=10, width=5, colors='k', labelsize=20)
axes[1].tick_params(direction='in', length=10, width=5, colors='k', labelsize=20)

axes[0].set_title('Actual', fontsize=18)
axes[1].set_title('Predicted', fontsize=18)
```



□ [실습] iris 군집화

- Scikit-learn 에 내장되어 있는 데이터셋 iris 이용
  - 군집결과 확인

```
iris_df = pd.DataFrame(iris.data, columns = iris.feature_names)
iris_df["target"] = pd.DataFrame(iris.target)
iris_df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0

```
iris_df["new_labels"] = new_labels
iris_df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	new_labels
0	5.1	3.5	1.4	0.2	0	0

```
iris_result = iris_df.groupby(["target", "new_labels"])["sepal length (cm)"].count()
iris_result
```

```
target  new_labels
0       0             50
1       1             12
        2             38
2       1             35
        2             15
Name: sepal length (cm), dtype: int64
```

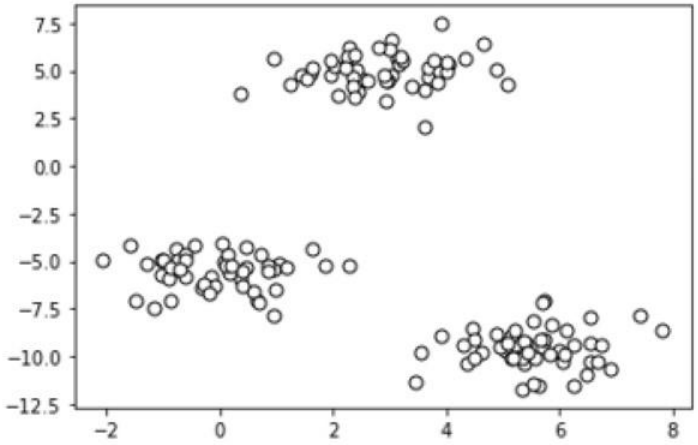
□ [실습] iris 군집화

- Scikit-learn 에 내장되어 있는 데이터셋 iris 이용
  - make_plobs() : 군집화용 데이터 생성

```
from sklearn.datasets import make_blobs
```

```
X, y = make_blobs(n_samples=150, n_features=2,
                  centers=3, random_state=10)
```

```
plt.scatter(X[:, 0], X[:, 1], c='white', marker='o', edgecolor='black', s=50)
plt.show()
```



□ [실습] iris 군집화

- Scikit-learn 에 내장되어 있는 데이터셋 iris 이용
  - make_plobs() : 군집화용 데이터 생성

```
km = KMeans(n_clusters=3, random_state=102)
km.fit(X)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=102, tol=0.0001, verbose=0)
```

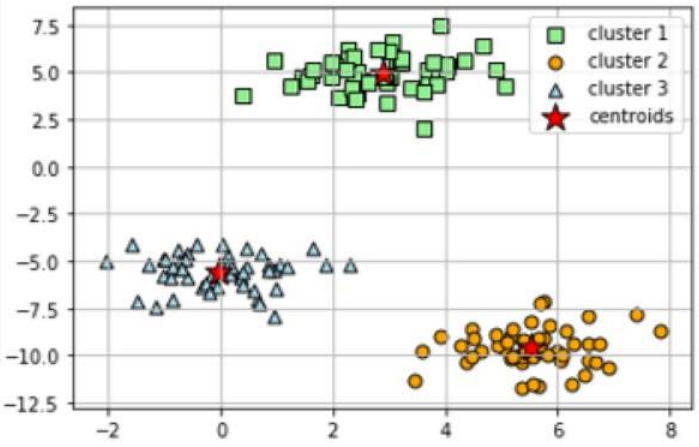
```
# 군집별 표시
plt.scatter(X[y_km == 0, 0], X[y_km == 0, 1],
            s=50, c='lightgreen', marker='s', edgecolor='black', label='cluster 1')

plt.scatter(X[y_km == 1, 0], X[y_km == 1, 1],
            s=50, c='orange', marker='o', edgecolor='black', label='cluster 2')

plt.scatter(X[y_km == 2, 0], X[y_km == 2, 1],
            s=50, c='lightblue', marker='^', edgecolor='black', label='cluster 3')

# 중심점 표시
plt.scatter(km.cluster_centers_[:, 0], km.cluster_centers_[:, 1],
            s=250, marker='*', c='red', edgecolor='black', label='centroids')

plt.legend()
plt.grid()
plt.show()
```



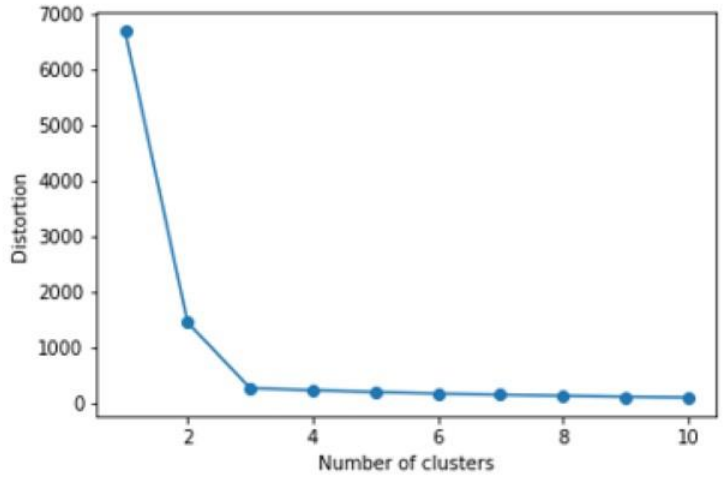
□ [실습] iris 군집화

- Scikit-learn 에 내장되어 있는 데이터셋 iris 이용
  - K값 결정

```
distortions = []

for i in range(1, 11):
    km = KMeans(n_clusters=i,random_state=102)
    km.fit(X)
    distortions.append(km.inertia_)

# plot
plt.plot(range(1, 11), distortions, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Distortion')
plt.show()
```

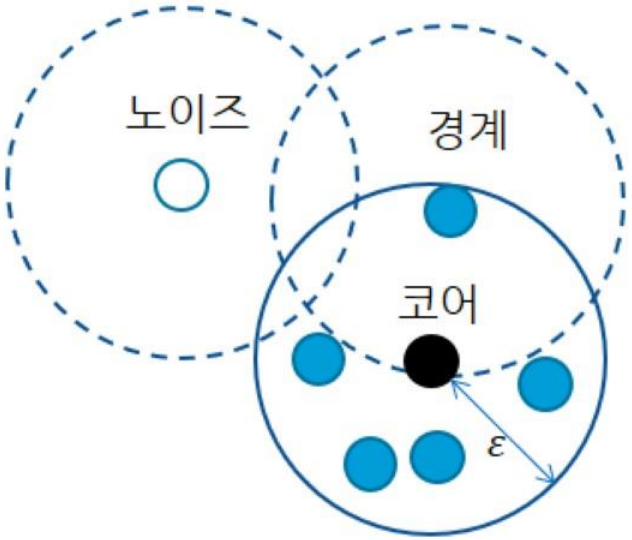


□ [과제] 와인데이터 군집화

- K-means 이용, 적절한 k 확인

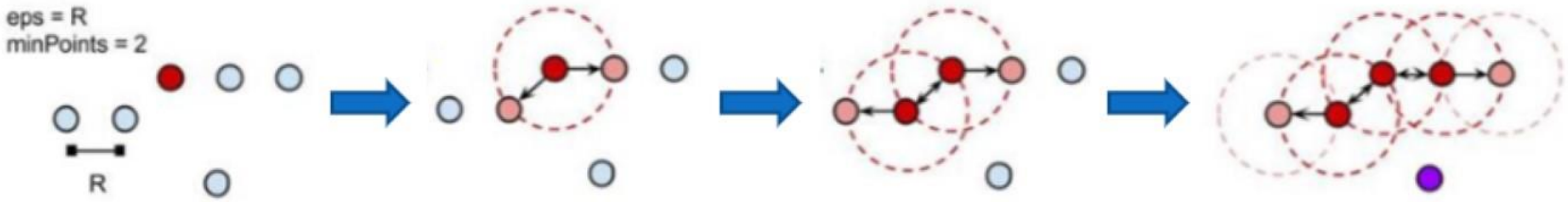
□ DBSCAN

- Density Based Spatial Clustering of Application with Noise
- 밀도있게 연결돼 있는 데이터 집합은 동일한 클러스터
- 용어
  - 밀도: 자기를 중심으로 반지름 ε 안에 있는 다른 좌표점의 개수
  - 최소 거리 ε : 이웃(neighborhood)을 정의하기 위한 거리, 밀도측정 반지름
  - 최소 데이터 갯수(minimum points): 밀집지역을 정의하기 위해 필요한 이웃의 갯수
    반지름 엡실론 내에 있는 최소 데이터의 개수

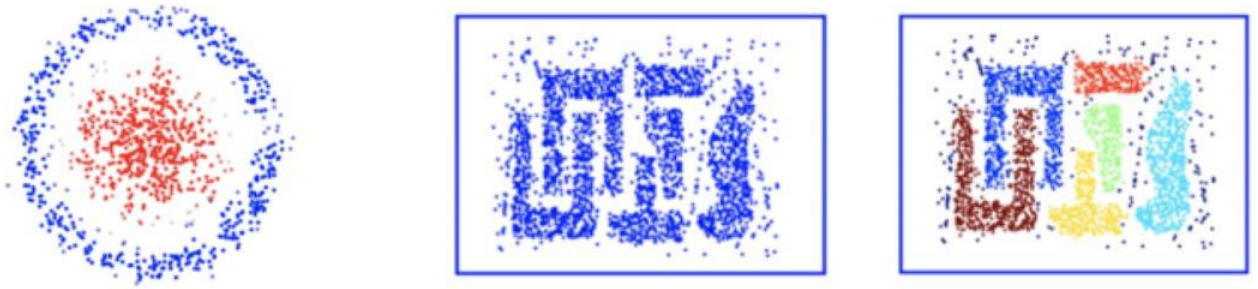


□ DBSCAN

- 군집화 과정



- 군집화 예



https://medium.com/@rdhawan201455/dbscan-clustering-algorithm-with-maths-c40dcee88281

□ DBSCAN

- 하이퍼파라미터 결정
  - DBSCAN은 군집 분석을 적용하고자 하는 데이터에 대한 이해도가 충분할 때 파라미터 설정이 쉬움
  - Minpts
    - 3이상으로 설정
    - 간단하게는, feature(차원)의 수 +1 을 기본값으로 설정
  - ε
    - ε은 반경을 설정하는 것으로
    - 너무작은 경우 많은 데이터가 이상치로 분류되며
    - 너무 큰 경우 군집 수가 하나가 될 가능성이 있음

□ DBSCAN

- 장점
  - k-means와 다르게 군집의 수를 설정할 필요가 없음
  - 다양한 모양의 군집이 형성될 수 있으며, 군집끼리 겹치는 경우가 없음
  - 노이즈 개념 덕분에 이상치에 대응이 가능
  - 설정할 파라미터가 두개(ε, MinPts)로 설정을 잘 한다면 좋은 성능을 낼 수 있음

- 단점
  - 한 데이터는 하나의 군집에 속하게 되므로 시작점에 따라 다른 모양의 군집이 형성됨
  - ε의 크기에 의해 성능이 크게 좌우됨
  - 군집별로 밀도가 다른경우 군집화가 제대로 이루어지지 않음

□ [실습] iris 군집화

- Scikit-learn 에 내장되어 있는 데이터셋 iris 이용
  - 데이터준비

```
from sklearn import datasets
```

```
iris = datasets.load_iris()
```

```
iris_df = pd.DataFrame(iris.data,
                       columns = ["sepal length", "sepal width",
                                  "petal length", "petal width"])
iris_df["labels"] = pd.DataFrame(iris.target)
iris_df.head()
```

	sepal length	sepal width	petal length	petal width	labels
0	5.1	3.5	1.4	0.2	0

```
feature = iris_df[["sepal length", "sepal width", "petal length", "petal width"]]
feature.head()
```

	sepal length	sepal width	petal length	petal width
0	5.1	3.5	1.4	0.2

□ [실습] iris 군집화

- Scikit-learn 에 내장되어 있는 데이터셋 iris 이용
  - 모델 생성

```
from sklearn.cluster import DBSCAN
import matplotlib.pyplot  as plt
import seaborn as sns
```

```
model = DBSCAN(eps=0.5,min_samples=5)
clst = model.fit_predict(feature)
```

  - 결과 확인

```
clst
```

```
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, -1,  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,  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,  1,  1,  1,  1,  1,  1],
      dtype=int64)
```

□ [실습] iris 군집화

- Scikit-learn 에 내장되어 있는 데이터셋 iris 이용
  - 결과 확인

```
iris_df["predict"] = predict
iris_df.head()
```

	sepal length	sepal width	petal length	petal width	labels	predict
0	5.1	3.5	1.4	0.2	0	0

```
iris_result = iris_df.groupby(["labels", "predict"])["sepal length"].count()
iris_result
```

```
labels  predict
0       -1          1
         0         49
1       -1          6
         1         44
2       -1         10
         1         40
Name: sepal length, dtype: int64
```

  - 시각화데이터 준비

```
predict = pd.DataFrame(clst)
predict.columns=['predict']
```

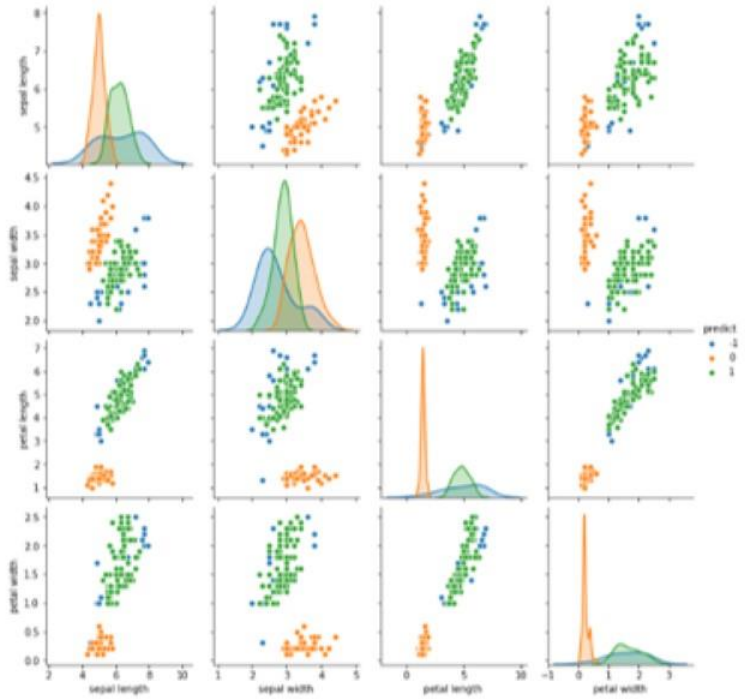
```
iris_pred = feature.join(predict)
iris_pred
```

	sepal length	sepal width	petal length	petal width	predict
0	5.1	3.5	1.4	0.2	0

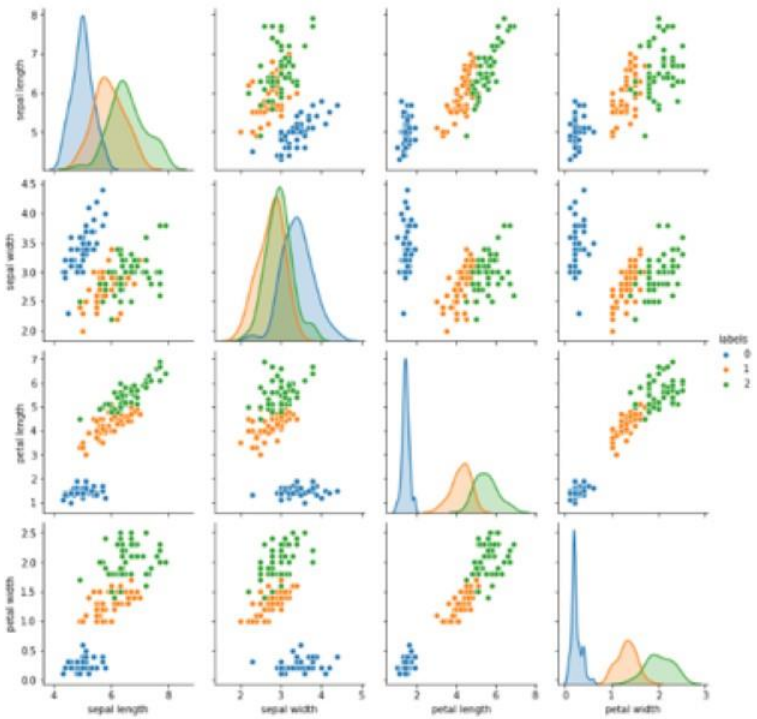
□ [실습] iris 군집화

- Scikit-learn 에 내장되어 있는 데이터셋 iris 이용
  - 시각화

```
sns.pairplot(iris_pred, hue='predict')
plt.show()
```




```
sns.pairplot(iris_df, hue='labels')
plt.show()
```

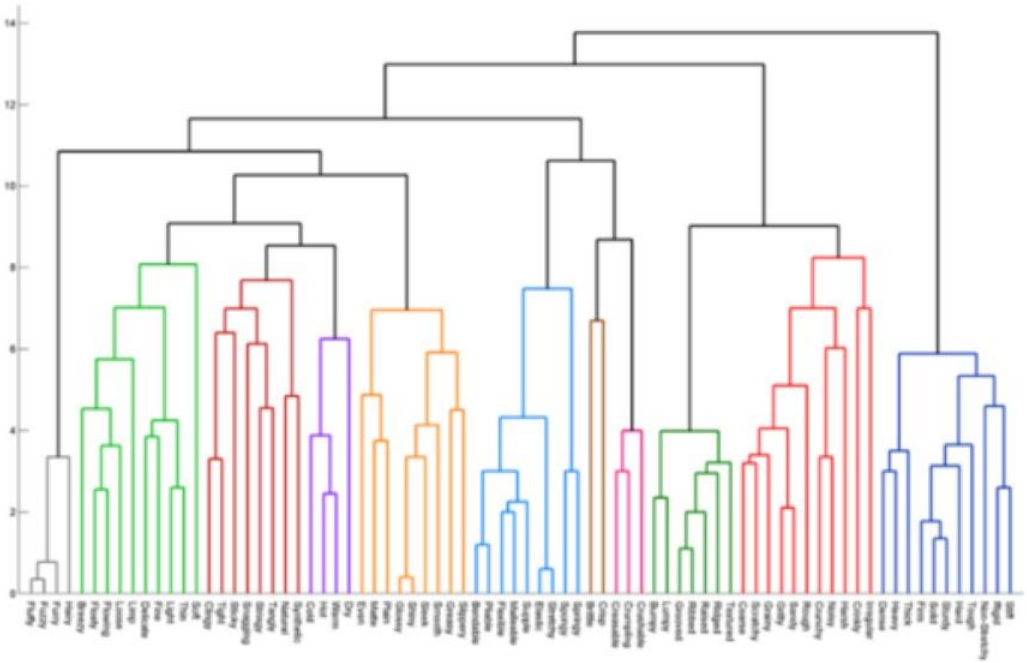


□ [과제] 와인데이터 군집화

- DBSCAN 이용

☐ Hierarchical Clustering

- 유사한 특성을 지닌 데이터를 묶어 이진 트리 형태(dendrogram)로 만들어가는 방법
- 초기에 클러스터의 개수를 미리 정할 필요 없음



https://ratsgo.github.io/machine%20learning/2017/04/18/HC/

☐ Hierarchical Clustering

- 유사한 특성을 지닌 데이터를 묶어 이진 트리 형태(dendrogram)로 만들어가는 방법
- 초기에 클러스터의 개수를 미리 정할 필요 없음

	A	B	C	D
A		20	7	2
B			10	25
C				3
D				

	AD	B	C	
AD		20	3	
B			10	
C				

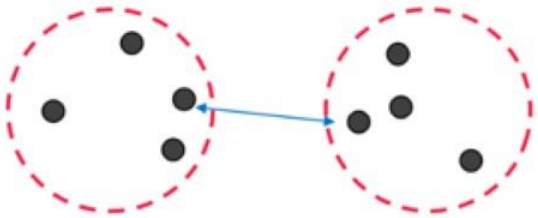
	ADC	B		
ADC		10		
B				

https://ratsgo.github.io/machine%20learning/2017/04/18/HC/

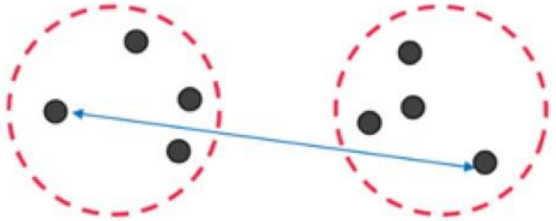
☐ Hierarchical Clustering

- 클러스터간 거리

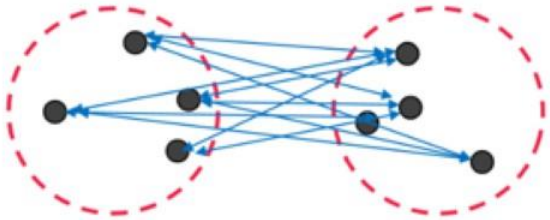
Min(Single Link)



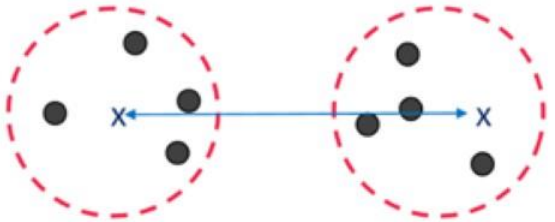
Max(Complete Link)



Average Link



Centroid



□ [실습] 계층적 군집 연습

- 데이터 생성, 확인

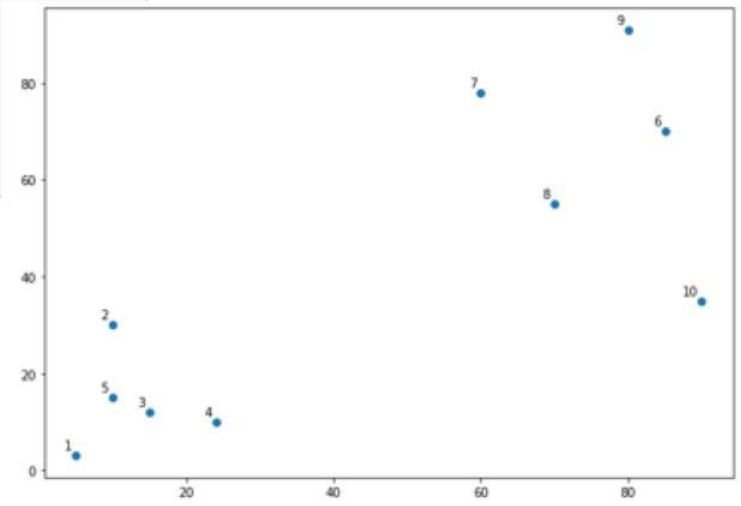
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

```
X = np.array([[5,3], [10,30], [15,12], [24,10], [10, 15],
              [85,70], [60,78], [70,55], [80,91], [90, 35]])
```

```
labels = range(1, 11)
plt.figure(figsize=(10, 7))

plt.scatter(X[:,0],X[:,1])

for label, x, y in zip(labels, X[:, 0], X[:, 1]):
    plt.annotate(label,
                 xy=(x, y), xytext=(-3, 3), textcoords='offset points',
                 ha='right', va='bottom')
plt.show()
```



□ [실습] 계층적 군집 연습

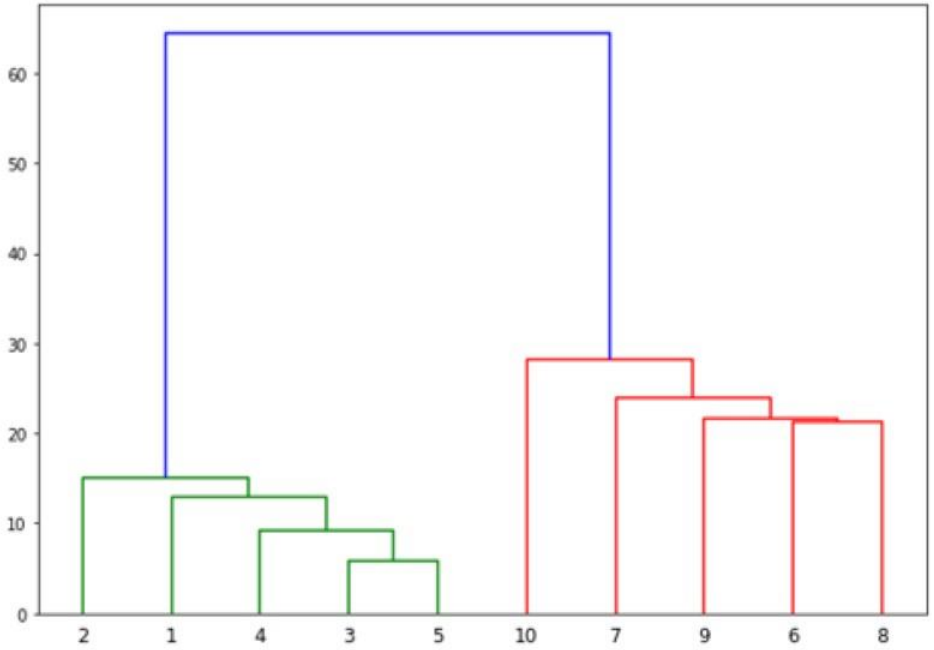
- 군집, 이진트리 생성

```
from scipy.cluster.hierarchy import dendrogram, linkage

linked = linkage(X, 'single')
```

```
labelList = range(1, 11)

plt.figure(figsize=(10,7))
dendrogram(linked, labels=labelList)
plt.show()
```



□ [실습] iris 군집화

- Scikit-learn 에 내장되어 있는 데이터셋 iris 이용
  - 데이터준비

```
from sklearn import datasets
```

```
iris = datasets.load_iris()
```

```
import pandas as pd
from sklearn import datasets
```

```
iris = datasets.load_iris()
```

```
feature = pd.DataFrame(iris.data,
                       columns = ["sepal length", "sepal width",
                                  "petal length", "petal width"])
feature.head()
```

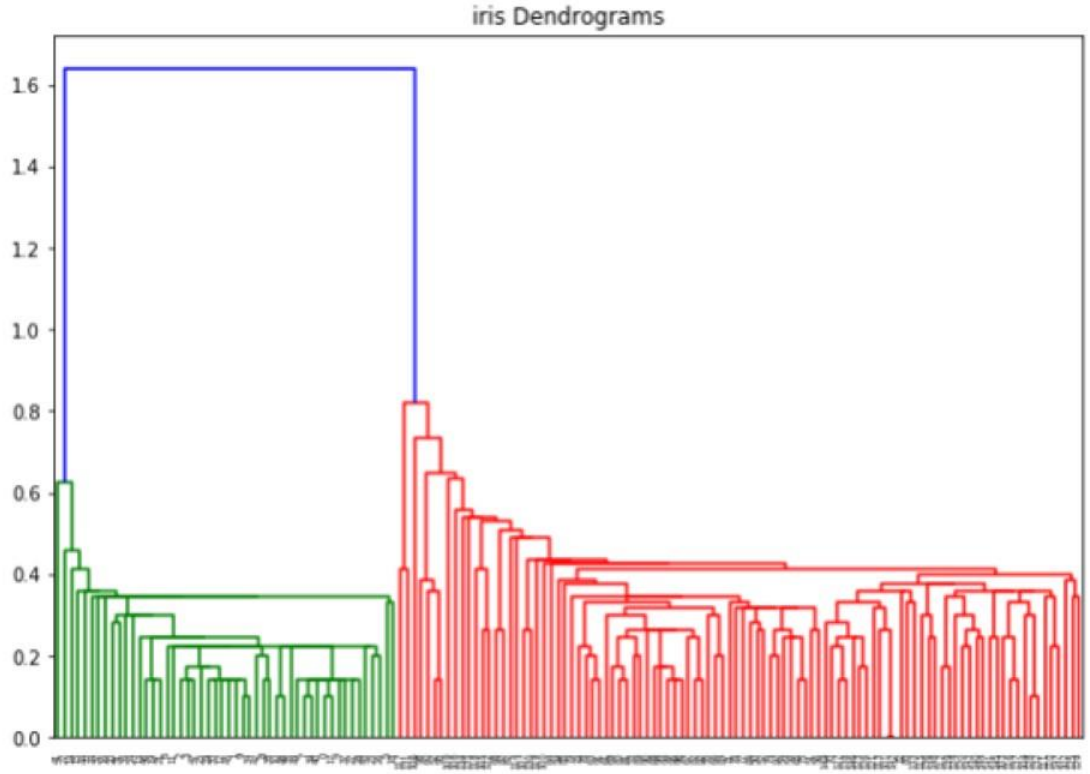
	sepal length	sepal width	petal length	petal width
0	5.1	3.5	1.4	0.2

□ [실습] iris 군집화

- Scikit-learn 에 내장되어 있는 데이터셋 iris 이용
  - 모델생성, 이진트리 생성

```
import scipy.cluster.hierarchy as shc

plt.figure(figsize=(10, 7))
plt.title("iris Dendrograms")
dend = shc.dendrogram(shc.linkage(feature))
```



□ [실습] iris 군집화

- Scikit-learn 에 내장되어 있는 데이터셋 iris 이용
  - 결과 확인

```
from sklearn.cluster import AgglomerativeClustering
```

```
cluster = AgglomerativeClustering(n_clusters=3)
cluster.fit_predict(feature)
```

```
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, 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, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2,
       2, 2, 2, 0, 0, 2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 0, 2, 2, 2,
       2, 0, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 0], dtype=int64)
```

  - 시각화데이터 준비

```
label = pd.DataFrame(cluster.labels_)
label.columns=['label']
```

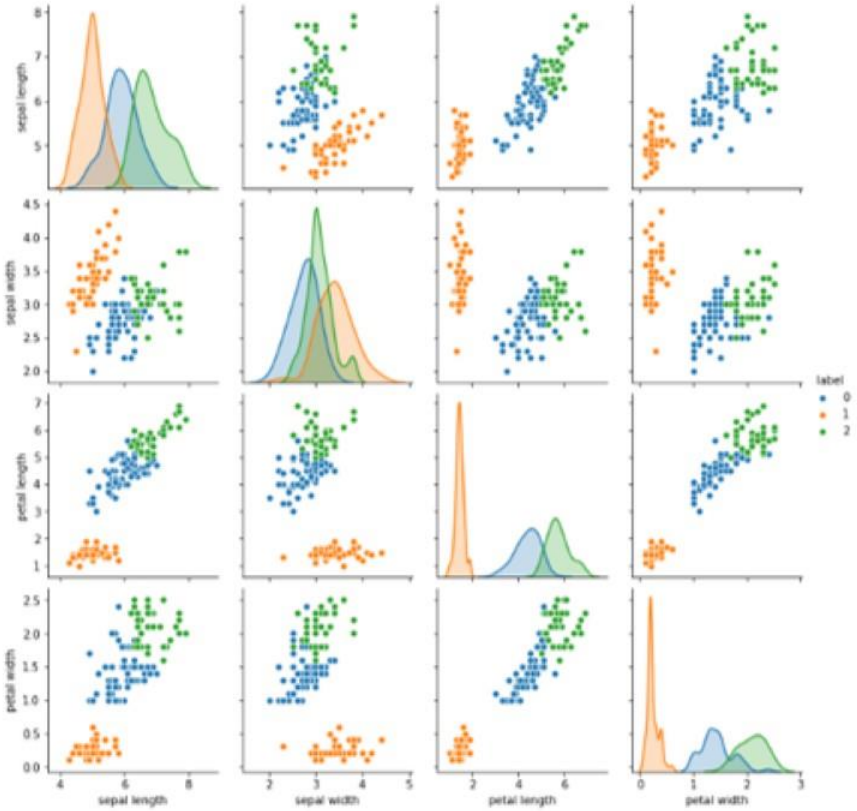
```
iris_pred = feature.join(label)
iris_pred
```

	sepal length	sepal width	petal length	petal width	label
0	5.1	3.5	1.4	0.2	1

□ [실습] iris 군집화

- Scikit-learn 에 내장되어 있는 데이터셋 iris 이용
  - 시각화

```
sns.pairplot(iris_pred, hue='label')
plt.show()
```



□ [과제] 와인데이터 군집화

- Hierarchical 이용