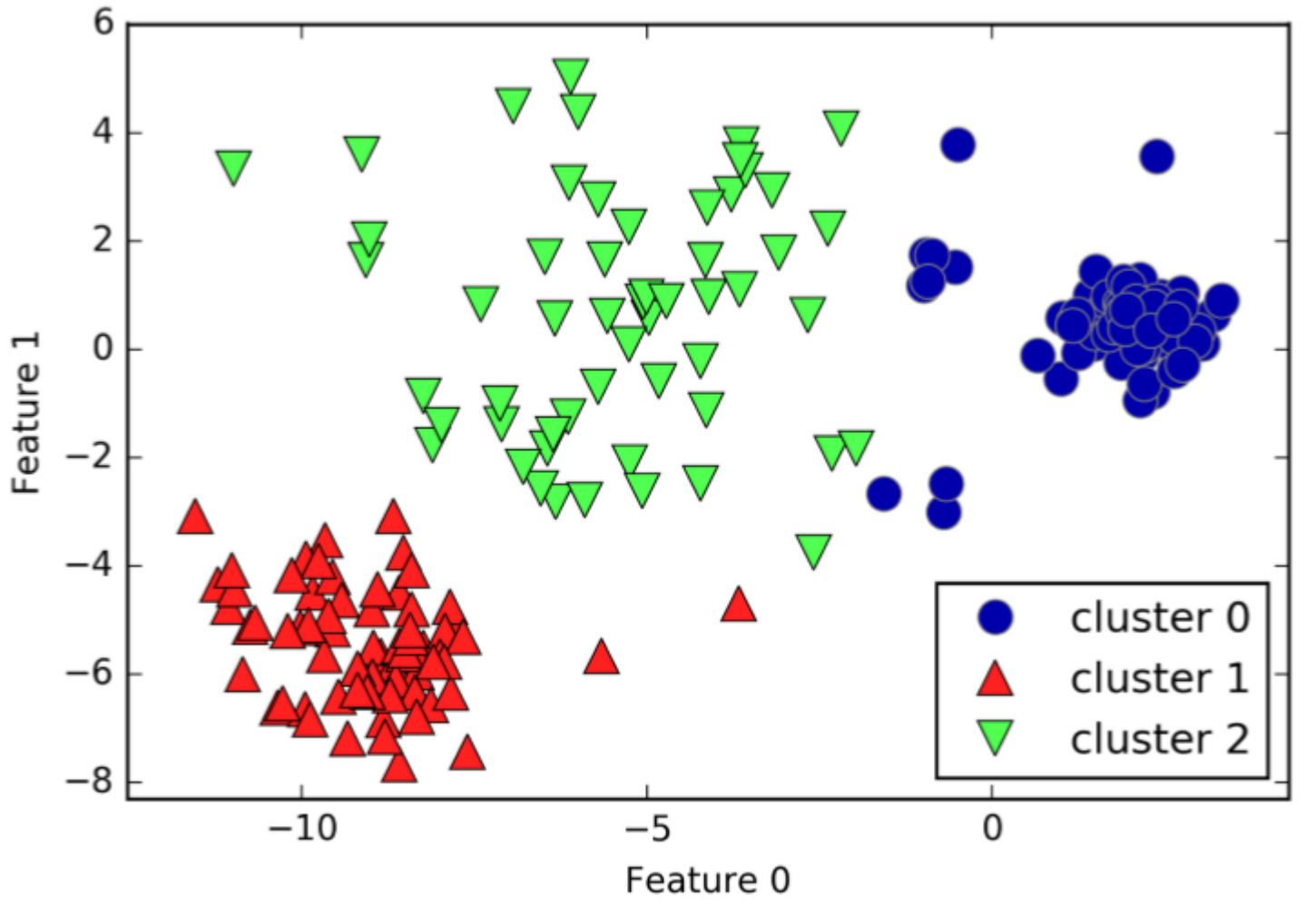
군집 분석 II

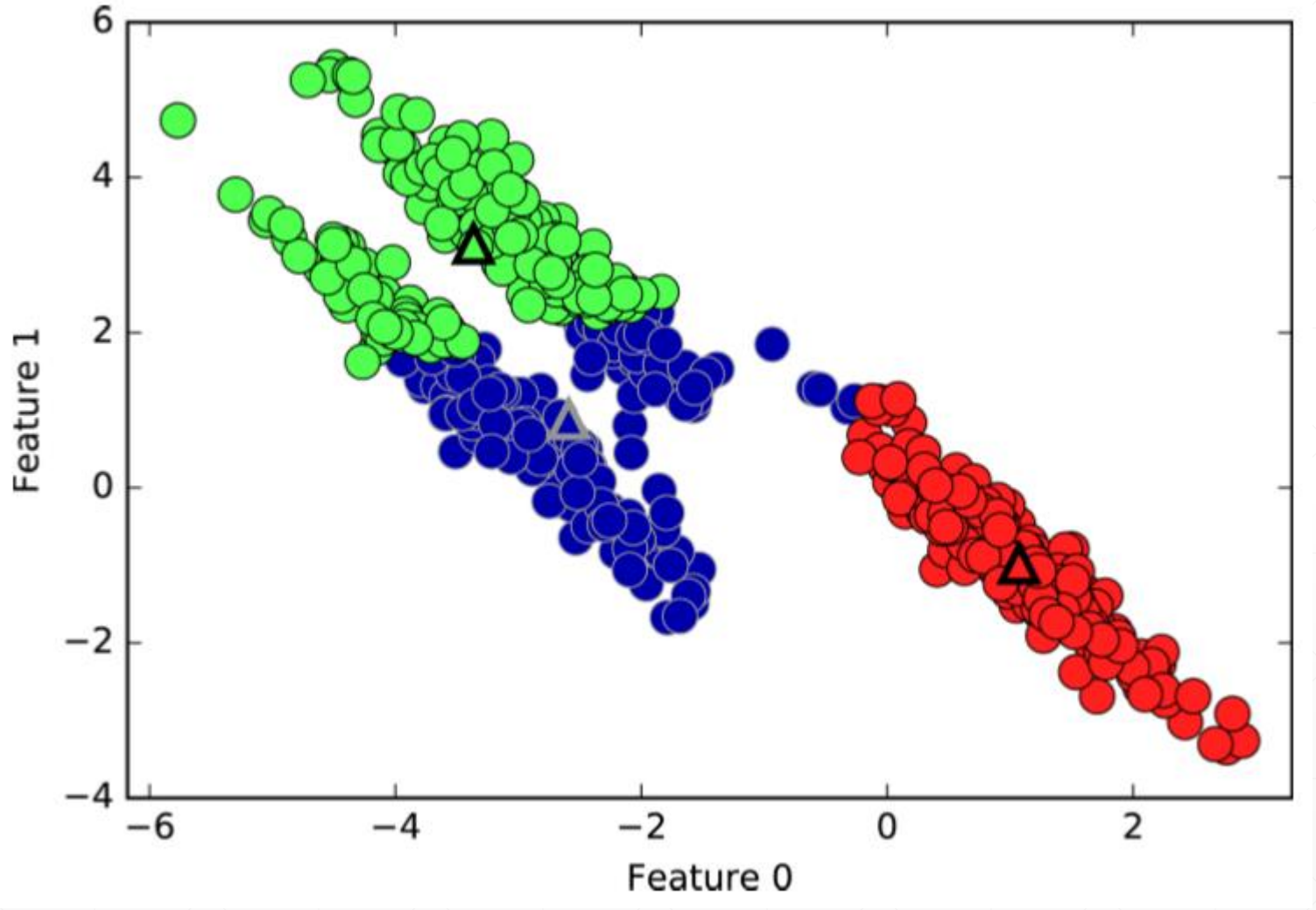
빅데이터 분석

Failure cases #1



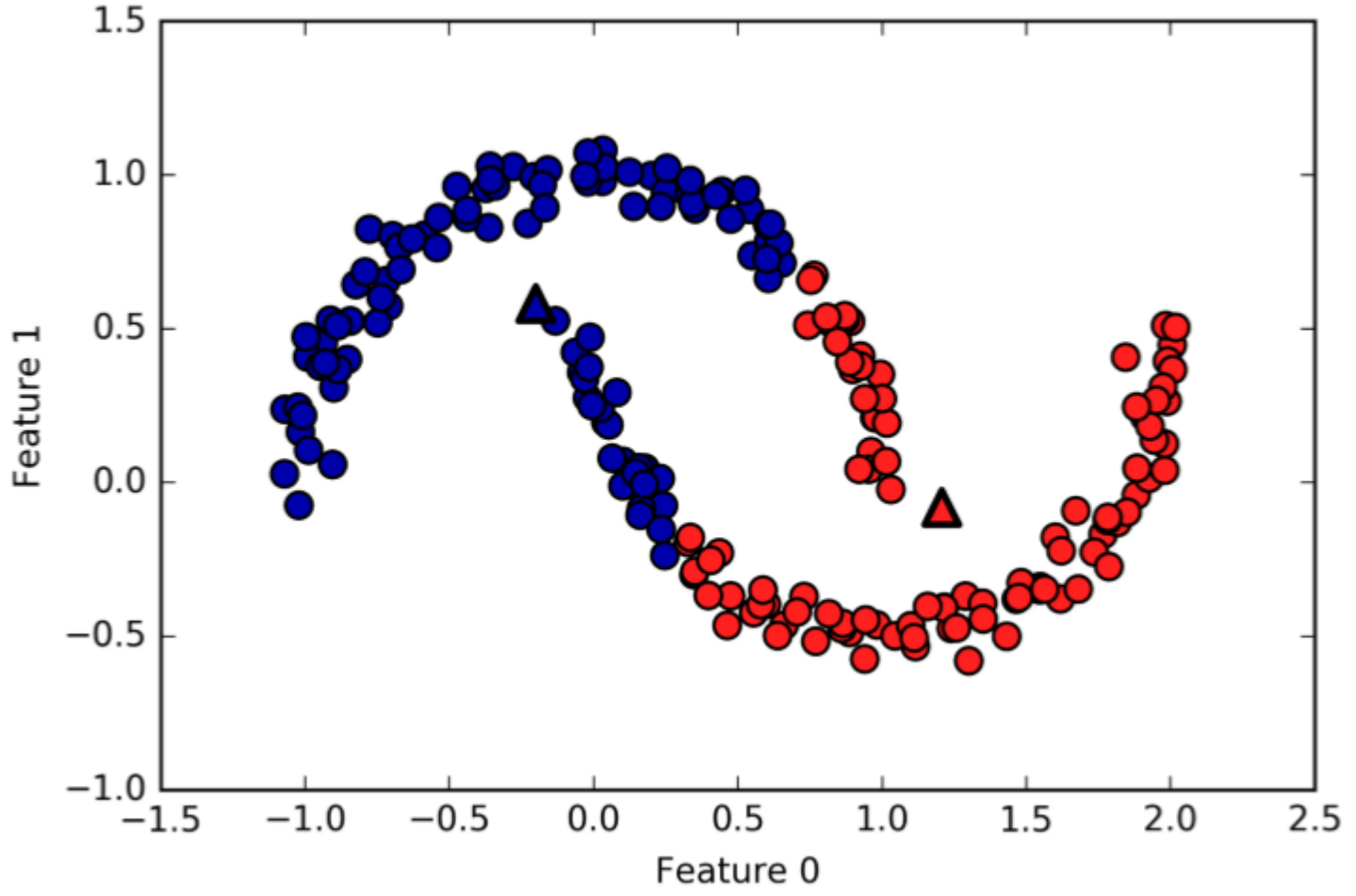
참고문헌 : Introduction to Machine Learning with Python
(by Andreas Muller, Sarah Guido)

Failure cases #2



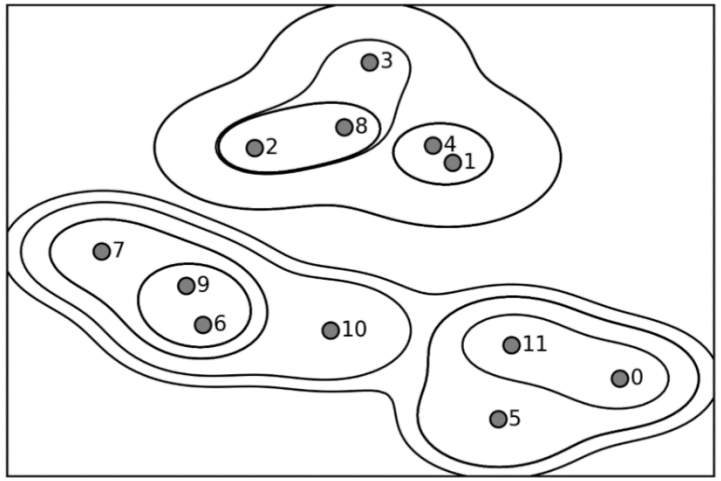
참고문헌 : Introduction to Machine Learning with Python
(by Andreas Muller, Sarah Guido)

Failure cases #3

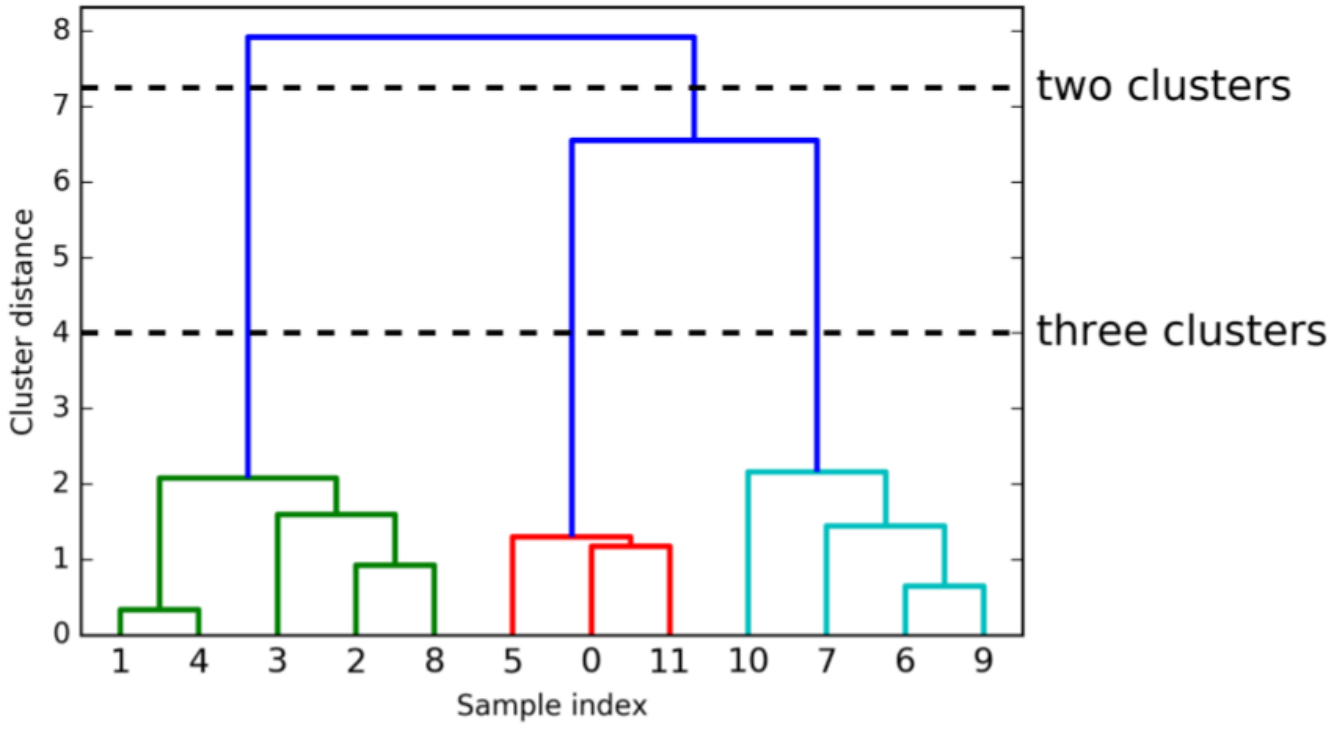


참고문헌 : Introduction to Machine Learning with Python
(by Andreas Muller, Sarah Guido)

Dendrogram



- **가지의 길이**는 합쳐진 클러스터가 얼마나 멀리 떨어져 있는지를 보여줍니다.
- 덴드로그램에서 가장 긴 가지는 “three clusters”로 표시한 점선이 가로지르는 세 개의 수직선입니다. 이 가지가 가장 길다는 것은 클러스터가 *세 개에서 두 개로 될 때 꽤 먼 거리의 포인트를 모은다는 뜻*입니다.

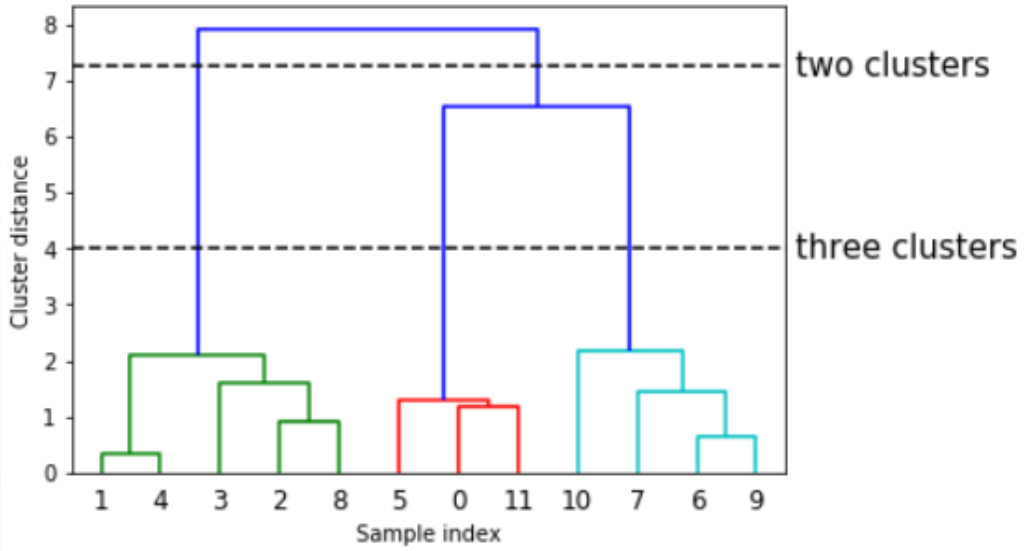


Dendrogram

참고문헌 : Introduction to Machine Learning with Python
(by Andreas Muller, Sarah Guido)

Dendrogram (cont'd)

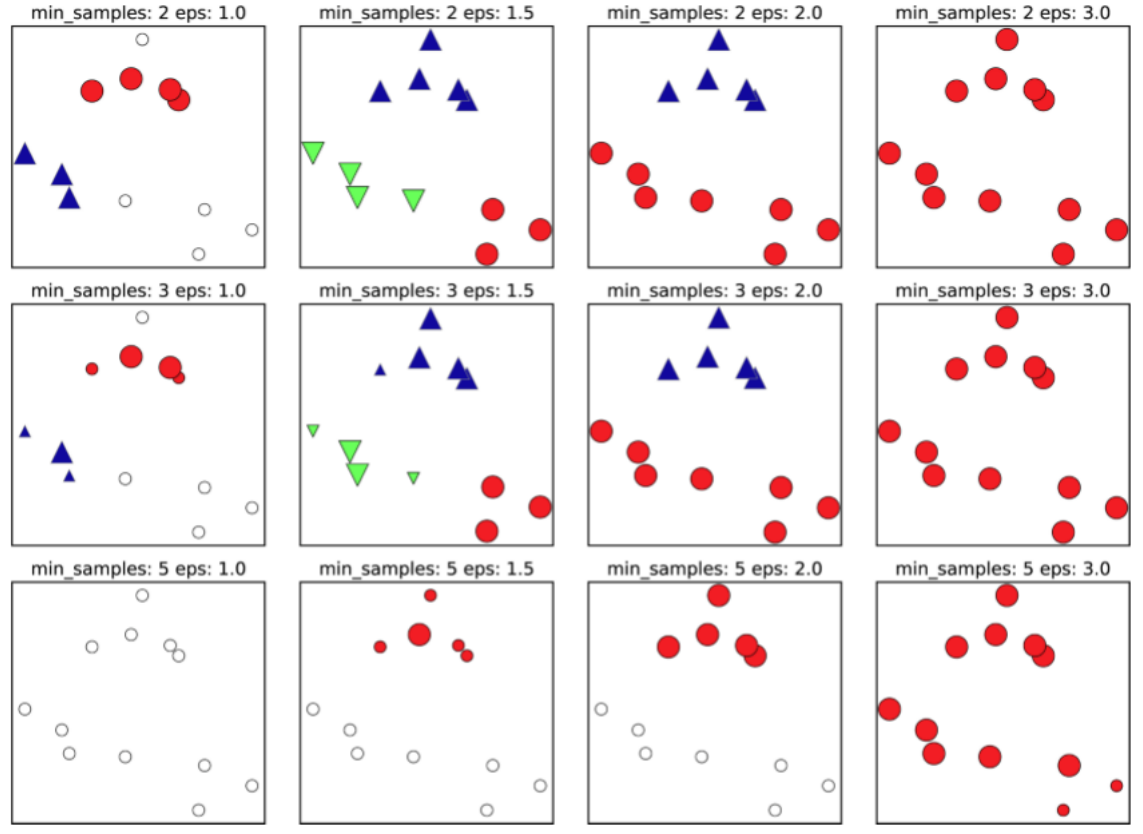
```
# Import the dendrogram function and the ward clustering function from SciPy
from scipy.cluster.hierarchy import dendrogram, ward

X, y = make_blobs(random_state=0, n_samples=12)
# Apply the ward clustering to the data array X
# The SciPy ward function returns an array that specifies the distances
# bridged when performing agglomerative clustering
linkage_array = ward(X)

# Now we plot the dendrogram for the linkage_array containing the distances
# between clusters
dendrogram(linkage_array)

# Mark the cuts in the three that signify two or three clusters
ax = plt.gca()
bounds = ax.get_xbound()
ax.plot(bounds, [7.25, 7.25], '--', c='k')
ax.plot(bounds, [4, 4], '--', c='k')

ax.text(bounds[1], 7.25, ' two clusters', va='center', fontdict={'size':15})
ax.text(bounds[1], 4, ' three clusters', va='center', fontdict={'size':15})
plt.xlabel("Sample index")
plt.ylabel("Cluster distance")
```



참고문헌 : Introduction to Machine Learning with Python
(by Andreas Muller, Sarah Guido)

DBSCAN

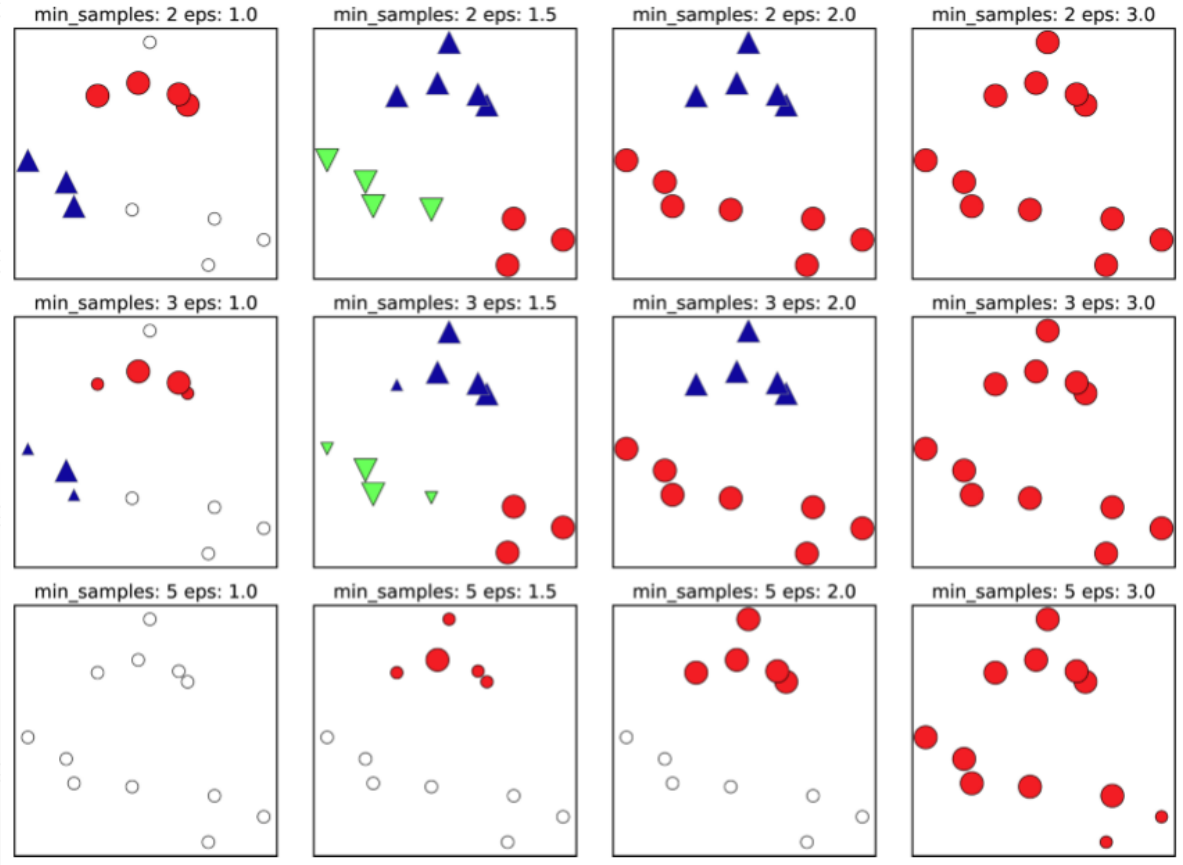
- DBSCAN(Density-Based Spatial Clustering of Applications with Noise) 장점 :
  1. 클러스터의 개수를 미리 지정할 필요가 없다는 점
  2. 복잡한 형상도 찾을 수 있으며
  3. 어떤 클래스에도 속하지 않는 포인트를 구분할 수 있습니다.
  4. 병합 군집이나 k-means보다는 다소 느리지만 비교적 큰 데이터셋에도 적용할 수 있습니다.



원리 : 데이터의 밀집 지역이 한 클러스터를 구성하며 비교적 비어있는 지역을 경계로 다른 클러스터와 구분된다는 것입니다.

참고문헌 : Introduction to Machine Learning with Python
(by Andreas Muller, Sarah Guido)

DBSCAN (cont'd)



- **밀집 지역**(dense region) : 특성 공간에서 가까이 있는 데이터가 많아 붐비는 지역을 의미함

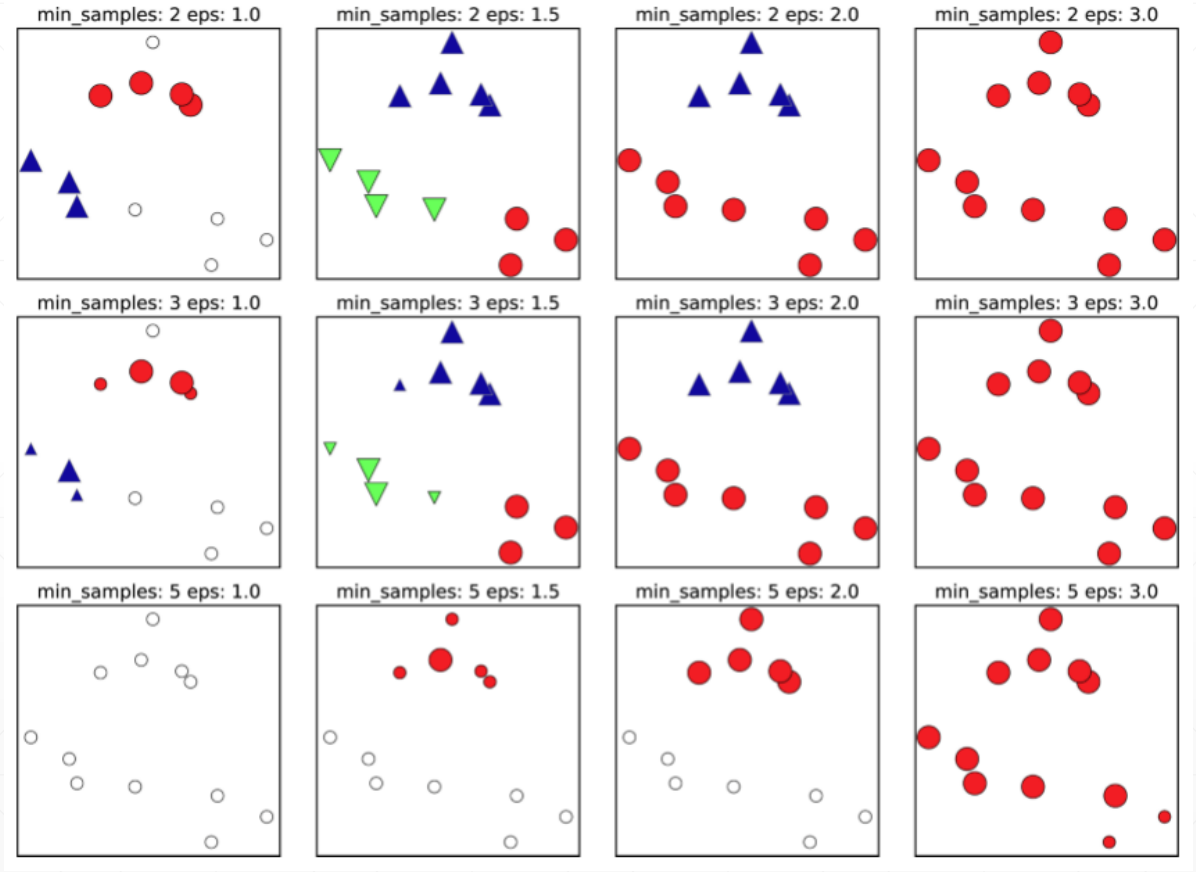
- **핵심 샘플(또는 핵심 포인트)** : 밀집 지역에 있는 포인트를 의미함
- **경계 포인트** : 핵심 포인트에서 eps 거리 안에 있는 포인트를 의미함
- **잡음 포인트**

알고리즘 :

1. 무작위로 포인트를 선택합니다.
2. 그 포인트에서 eps 거리 안의 모든 포인트를 찾습니다. 만약 eps 거리 안에 있는 포인트 수가 min_samples보다 적다면 그 포인트는 어떤 클래스에도 속하지 않는 잡음으로 레이블합니다. eps 거리 안에 min_samples보다 많은 포인트가 있다면 그 포인트는 핵심 샘플로 레이블하고 새로운 클러스터 레이블을 할당합니다.
3. 그 포인트의(eps 거리 안의) 모든 이웃을 살핍니다.
4. 만약 핵심 샘플이면 그 포인트의 이웃을 차례로 방문합니다.
5. 이런 식으로 계속 진행하여 클러스터는 eps 거리 안에 더 이상 핵심 샘플이 없을 때까지 자라납니다.
6. 아직 방문하지 못한 포인트를 선택하여 같은 과정을 반복합니다.

참고문헌 : Introduction to Machine Learning with Python
(by Andreas Muller, Sarah Guido)

DBSCAN (cont'd)



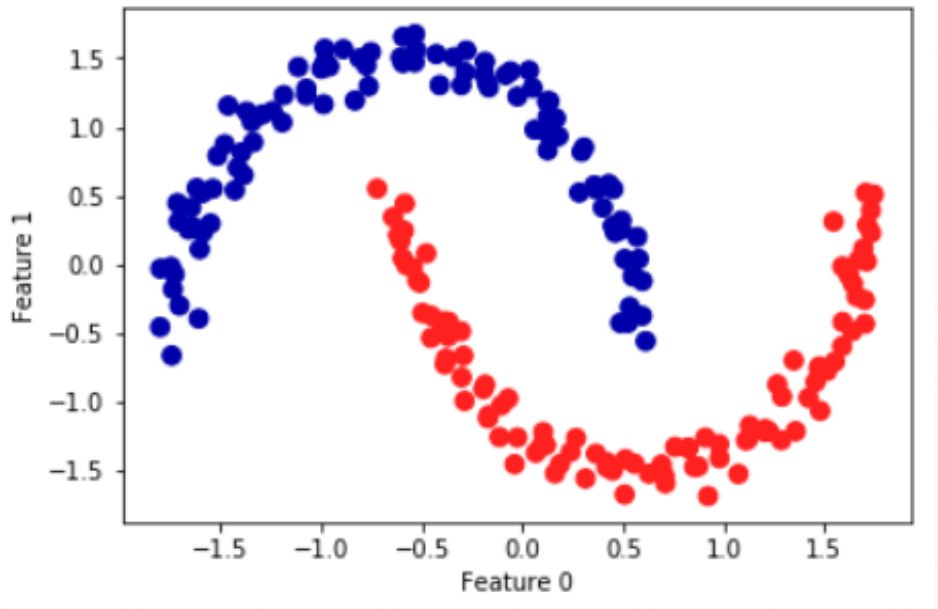
Parameters :

- eps를 증가시키면 하나의 클러스터에 더 많은 포인트가 포함됩니다. 이는 클러스터를 커지게 하지만 여러 클러스터를 하나로 합치게도 만듭니다.
- min_samples를 키우면 핵심 포인트 수가 줄어들며 잡음 포인트가 늘어납니다.

참고문헌 : Introduction to Machine Learning with Python (by Andreas Muller, Sarah Guido)

DBSCAN (cont'd)

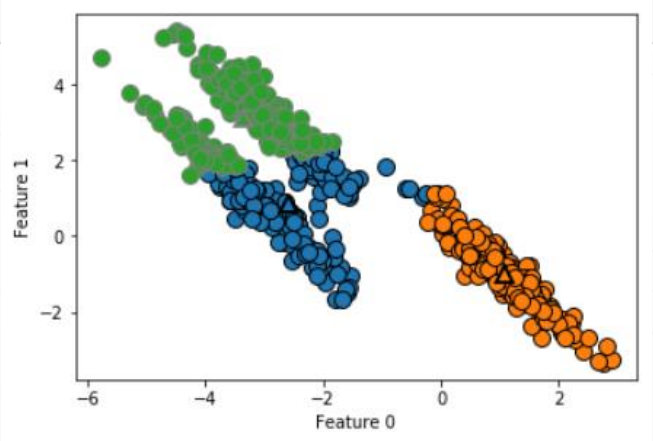
```
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_moons
from sklearn.cluster import DBSCAN
X, y = make_moons(n_samples=200, noise=0.05, random_state=0)

# rescale the data to zero mean and unit variance
scaler = StandardScaler()
scaler.fit(X)
X_scaled = scaler.transform(X)

dbscan = DBSCAN()
clusters = dbscan.fit_predict(X_scaled)
# plot the cluster assignments
plt.scatter(X_scaled[:,0], X_scaled[:,1], c=clusters, cmap=mglearn.cm2, s=60)
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
```



참고문헌 : Introduction to Machine Learning with Python
(by Andreas Muller, Sarah Guido)

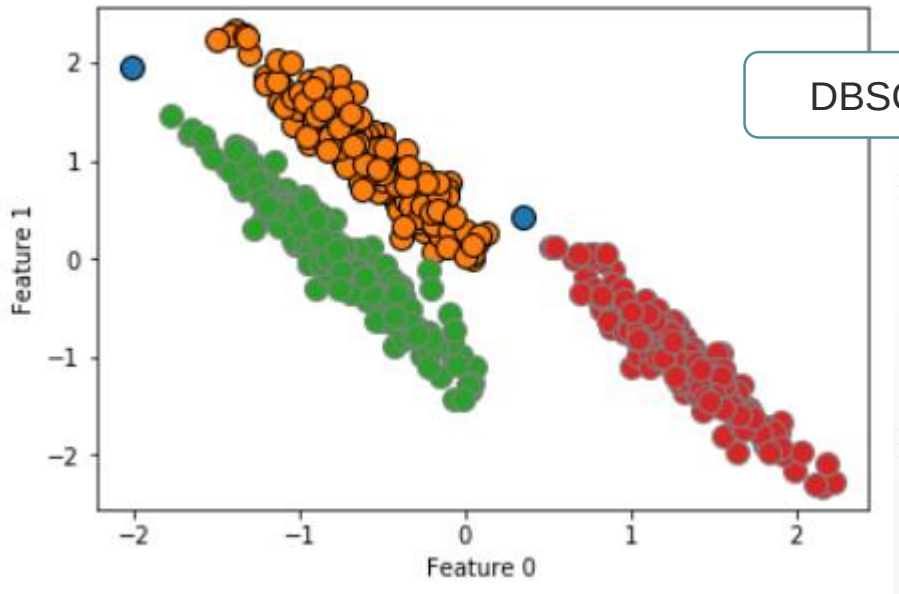
Comparison

```python
# generate some random cluster data
X, y = make_blobs(random_state=170, n_samples=600)
rng = np.random.RandomState(74)

# transform the data to be stretched
transformation = rng.normal(size=(2, 2))
X = np.dot(X, transformation)

# cluster the data into three clusters
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
y_pred = kmeans.predict(X)

# plot the cluster assignments and cluster centers
mglearn.discrete_scatter(X[:,0], X[:,1], kmeans.labels_, markers='o')
mglearn.discrete_scatter(kmeans.cluster_centers_[:,0],
                         kmeans.cluster_centers_[:,1],
                         [0,1,2], markers='^', markeredgewidth=2)
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
```



K-means

```python
# rescale the data to zero mean and unit variance
scaler = StandardScaler()
scaler.fit(X)
X_scaled = scaler.transform(X)

dbscan = DBSCAN(min_samples=3, eps=0.2)
clusters = dbscan.fit_predict(X_scaled)
# plot the cluster assignments and cluster centers
mglearn.discrete_scatter(X_scaled[:,0], X_scaled[:,1], clusters, markers='o')
plt.xlabel("Feature 0")
plt.ylabel("Feature 1")
```



DBSCAN

참고문헌 : Introduction to Machine Learning with Python
(by Andreas Muller, Sarah Guido)

Import Libraries

```
from IPython.display import display
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import mglearn
```

기본적인 라이브러리

```
from sklearn.datasets import load_breast_cancer

cancer = load_breast_cancer()
```

Cancer 데이터

```
from sklearn.model_selection import train_test_split
```

Train / Test 데이터

참고문헌 : Introduction to Machine Learning with Python
(by Andreas Muller, Sarah Guido)