Chapter 1	machine Learning with Scikit-Learn	분류 성능 평가

회귀 분석과 달리 모수에 대한 t-검정, 신뢰 구간(confidence interval) 추정 등이 쉽지 않기 때문에 이를 보완하기 위해 다양한 성능 평가 기준이 필요하다

	Positive라고 예측	Negative라고 예측
실제 Positive	True Positive	False Negative
실제 Negative	False Positive	True Negative

참고) DS(Fraud Detection System)의 예

FDS(Fraud Detection System)는 금융 거래, 회계 장부 등에서 잘못된 거래, 사기 거래를 찾아내는 시스템을 말한다. FDS 의 예측 결과가 Positive 이면 사기 거래라고 예측한 것이고 Negative 이면 정상 거래라고 예측한 것이다. 이 결과가 사실과 일치하는지 틀리는 지에 따라 다음과 같이 말한다.

True Positive: 사기를 사기라고 정확하게 예측
True Negative: 정상을 정상이라고 정확하게 예측
False Positive: 정상을 사기라고 잘못 예측
False Negative: 사기를 정상이라고 잘못 예측

Precision 정밀도

클래스에 속한다고 출력한 샘플 중 실제로 클래스에 속하는 샘플 수의 비율
FDS 의 경우, 사기 거래라고 판단한 거래 중 실제 사기 거래의 비율. 유죄율

precision = TP/ (TP+FP)

Recall 재현율

TPR: true positive rate
실제 클래스에 속한 샘플 중에 클래스에 속한다고 출력한 샘플의 수
FDS 의 경우, 실제 사기 거래 중에서 실제 사기 거래라고 예측한 거래의 비율.
검거율

recall = TP/(TP+FN)

F Score 정밀도(Precision)과 재현율(Recall)의 가중 조화 평균

F Score = 2 x precision x recall/(precision + recall)

Accuracy 정확도

accuracy = (TP+TN)/(TP+TN+FP+FN)

Chapter 1	machine Learning with Scikit-Learn	모델 저장

[실행할 때마다 학습하지 않도록 학습시킨 모델 저장.]

chapter01_sklearn/ex10_save_sklearn.py

```
from sklearn.linear_model import LinearRegression
from sklearn.externals import joblib

x = [[10],[5],[9],[7]]      #공부시간 10시간   5시간, 9시간, 7시간
y = [[100],[50],[90],[77]]  #시험점수 100점    50점,  90점   77점
model = LinearRegression()
model = model.fit(x, y)
result =model.predict([[7]])
print(result)

#파일에 학습 결과 저장
joblib.dump(model,filename="jumsu.pkl")   #pickle 파일
```

학습결과가 jumsu.pkl 파일로 저정됨

[저장한 모델 로드]

저장한 파일에서 학습 모델 로드

chapter01_sklearn/ex11_load_sklearn.py

```
from sklearn.externals import joblib

model= joblib.load(filename="jumsu.pkl")
result =model.predict( [[7] ])

print(result)
```

Chapter 1	machine Learning with Scikit-Learn	군집

비지도 학습

- 군집, 비지도 차원 축소 등.

군집 (Clustering) : 비슷한 특징을 가지는 데이터 인스턴스들끼리 그룹화.

비지도 차원 축소 :

대표적 예) 시각화를 위해 데이터셋을 2차원으로 변경

[군집 (Clustering)]

K Means 알고리즘

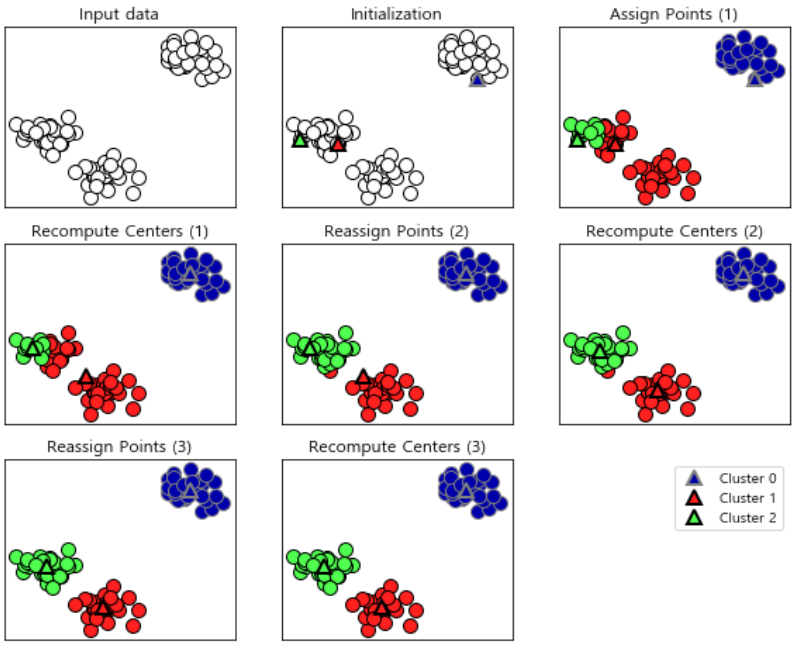
: 가장 간단하고 널리 알려진 군집 알고리즘. 클러스터의 중심을 찾는 알고리즘.

데이터 포인트를 가장 가까운 클러스터 중심에 할당하고, 거리 평균으로 클러스터 중심을 다시 지정합니다.

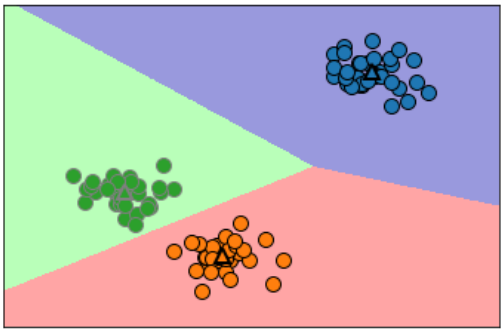
참고)

chapter01_sklearn/ex12.py K-Means 알고리즘 설명 그림

```
import mglearn
import matplotlib.pyplot as plt

#Kmeans 알고리즘 동작 설명 그림
mglearn.plots.plot_kmeans_algorithm()
plt.show()

# 경계선 그리기
mglearn.plots.plot_kmeans_boundaries()
plt.show()
```



클러스터들의 중심점을 가장 가까운 거리의 중심점이 될 수 있도록 반복하여 이동



[K Means 알고리즘으로 군집 예]

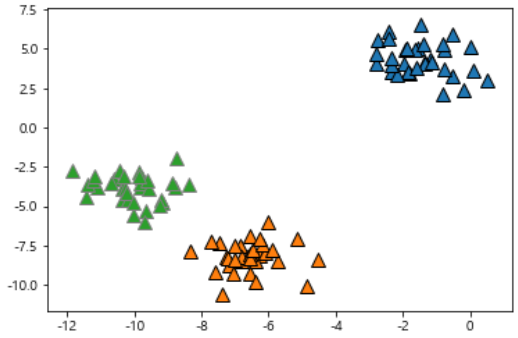
chapter01_sklearn/ex13_kmeans.py 예) 인위적인 2차원 데이터 생성 후, K Means 알고리즘으로 군집 (Clustering)

```
import mglearn
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import matplotlib
from matplotlib import font_manager, rc

#한글 폰트 등록
font_location = "c:/Windows/fonts/malgun.ttf"
font_name = font_manager.FontProperties(fname=font_location).get_name()
matplotlib.rc('font', family=font_name)

# 인위적으로 2차원 데이터를 생성합니다
X,y = make_blobs(random_state=1)
print(X)

# KMeans 3개의 클러스터로 군집화
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)

print(kmeans.labels_) # 클러스터링한 결과 라벨

mglearn.discrete_scatter(X[:,0], X[:,1], kmeans.labels_, markers='^')
# 세번째 매개인자인  클러스터링한 결과 라벨 (kmeans.labels_) 별로 다른 색 그래프
# x축 X[:,0],  y축 X[:,1]
plt.show()
```

출력 결과

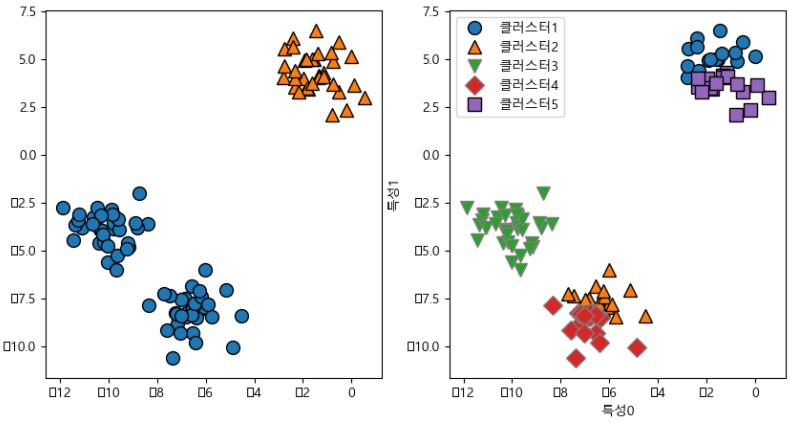
[[-7.04747278e+00 -9.27524683e+00] [-1.37397258e+00 5.29163103e+00] [-6.25393051e+00 -7.108... 중략]

[0 2 2 2 1 1 1 2 0 0 2 2 1 0 1 1 1 0 ... 중략]



chapter01_sklearn/ex13_kmeans. py (계속) 클러스터링 개수 변경

```
##############################################################################
fig, axes = plt.subplots(1,2,figsize=(10,5))   #1행 2열 sub그래프

#두 개의 클러스터 중심을 사용
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)
assignments = kmeans.labels_
mglearn.discrete_scatter(X[:,0],X[:,1],assignments, ax = axes[0])

##############################################################################
#다섯 개의 클러스터 중심
kmeans = KMeans(n_clusters=5)
kmeans.fit(X)
assignments = kmeans.labels_

mglearn.discrete_scatter(X[:,0],X[:,1],assignments, ax = axes[1])

plt.legend(["클러스터1","클러스터2","클러스터3","클러스터4","클러스터5"])
plt.ylabel("특성1")
plt.xlabel("특성0")
plt.show()
```



사례 1-1)

보습학원 원장인 김지영씨는 초등학생 아이들을 가르치는 학원을 운영하고 있습니다. 최근 새학기에 접어 들면서 학원생들의 인원수가 늘어남에 따라 가장 효율적으로 클래스를 운영하는 문제를 고민하게 되었습니다. 이왕이면 비슷한 성향의 학생들끼리 서로 묶어서 가르치게 되면 서로 친해지기도 쉽고, 시너지 효과도 나지 않을까? 아마도 이런 아이들이 좋아하는 선생님의 성향도 비슷해서 새로운 강사를 뽑을 때에 참고할 수 있지 않을까?

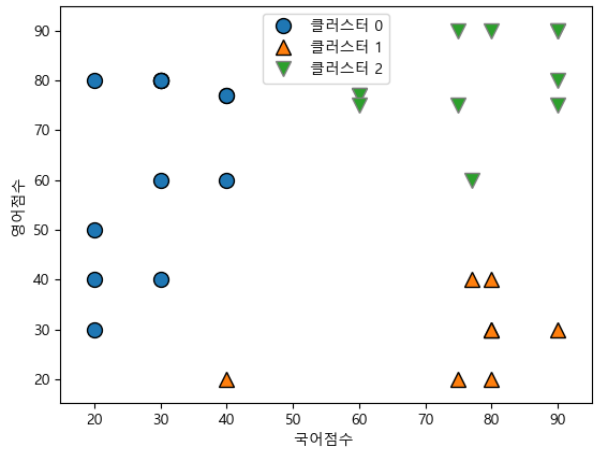
고민 끝에 자신이 가르치는 모든 학원생들에게 설문조사를 실시하여, 학생들에 대한 모든 속성 값을 조사했습니다. 설문 조사 결과는 다음과 같습니다. (파일명 ex14_academy.csv)

학번	국어점수	영어점수
1	90	80
2	90	75
3	90	90
4	80	20
5	80	30
6	80	30
7	77	40

.. 중략 .

chapter01_sklearn/ex14.py

```
import mglearn
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import matplotlib
from matplotlib import font_manager, rc

font_location = "c:/Windows/fonts/malgun.ttf"
font_name = font_manager.FontProperties(fname=font_location).get_name()
matplotlib.rc('font', family=font_name)

data =pd.read_csv("ex14_academy.csv" )
# 군집 모델을 만듭니다
kmeans = KMeans(n_clusters=3)
kmeans.fit(data.iloc[:,1:])

print("클러스터 레이블 ", kmeans.labels_)

mglearn.discrete_scatter(data.iloc[:,1], data.iloc[:,2], kmeans.labels_)
plt.legend(["클러스터 0", "클러스터 1", "클러스터 2"], loc='best')
plt.xlabel("국어점수 ")
plt.ylabel("영어점수 ")
plt.show()
```



문제) 국어점수 100, 영어점수 80 인 새로운 학생이 입학하였습니다. 이 학생은 몇 번 클러스터에 포함되어야 합니까?

사례 1-2)

보습학원 원장인 김지영씨는 초등학생 아이들을 가르치는 학원을 운영하고 있습니다. 최근 학부형들의 요청에 따라 학원의 과목을 영어와 국어 두가지 과목에서 수학, 과학을 추가하여 5가지 과목으로 확장하기로 하였습니다. 학생들의 영어,국어,수학,과학 과목의 점수 및 학업 성취도에 따라 비슷한 성향의 학생들끼리 서로 묶어서 3반으로 나누어 가르치려고 합니다. 학생들의 점수 조사 결과는 다음과 같습니다. (파일명 ex15_academy.csv)

.. 중략 .

학생번호	국어점수	영어점수	수학점수	과학점수	학업성취도
0	90	80	80	80	80
1	90	75	75	75	75
2	65	90	90	90	90
3	90	80	80	80	80
4	90	75	75	75	75

.. 중략 .

chapter01_sklearn/ex15.py

```
import pandas as pd
from sklearn.cluster import KMeans

data =pd.read_csv("ex15_academy.csv")

# 군집 모델을 만듭니다
kmeans = KMeans(n_clusters=3)
kmeans.fit( data.iloc[:,1:]  ) #학생번호 컬럼은 제외하고 학습

for no, cla in enumerate(kmeans.labels_):
    print("학생번호: {} : {}".format(no, cla))
```

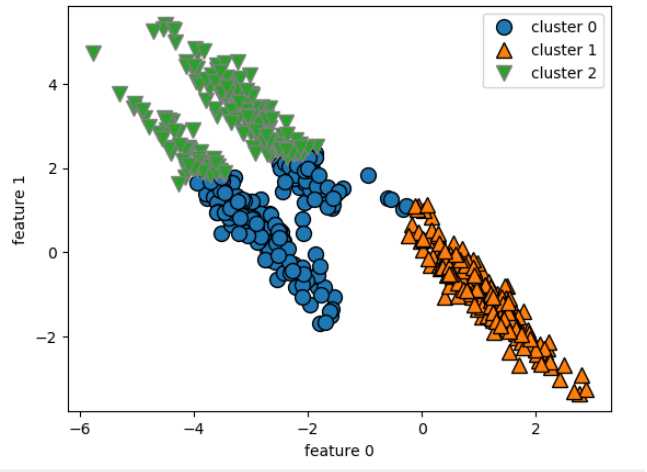
출력 결과

학생번호: 0, : 0
학생번호: 1, : 2
학생번호: 2, : 1
등....

[K Means 알고리즘이 실패하는 경우]

chapter01_sklearn/ex16.py 원형이 아닌 클러스터를 구분하지 못하는 K-Means 알고리즘

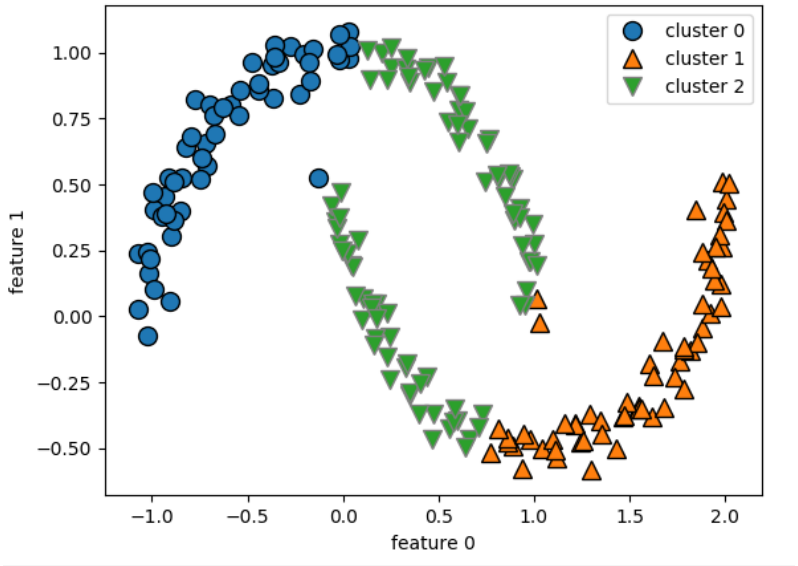
```
import mglearn
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np

#인위적인 2차원 데이터 셋 생성
X_varied, y_varied = make_blobs(n_samples=600,random_state=170)

#데이터가 길게 늘어지도록 변경
rng = np.random.RandomState(74)
transformation = rng.normal(size = (2,2))
X_varied = np.dot(X_varied, transformation)

# 3개의 클러스터 생성
y_pred = KMeans(n_clusters=3).fit_predict(X_varied)

mglearn.discrete_scatter(X_varied[:,0], X_varied[:,1], y_pred)
plt.legend(["cluster 0","cluster 1","cluster 2"],loc="best")
plt.xlabel("feature 0")
plt.ylabel("feature 1")
plt.show()
```



각 클러스터를 정의하는 것이 중심 하나 뿐입니다. 클러스터는 둥근 형태로 나타납니다. 모든 클러스터의 반경이 똑같다고 가정하고, 모든 방향이 똑같이 중요하다고 가정합니다.

chapter01_sklearn/ex17.py 더 복잡한 형태라면 성능이 더 나빠집니다

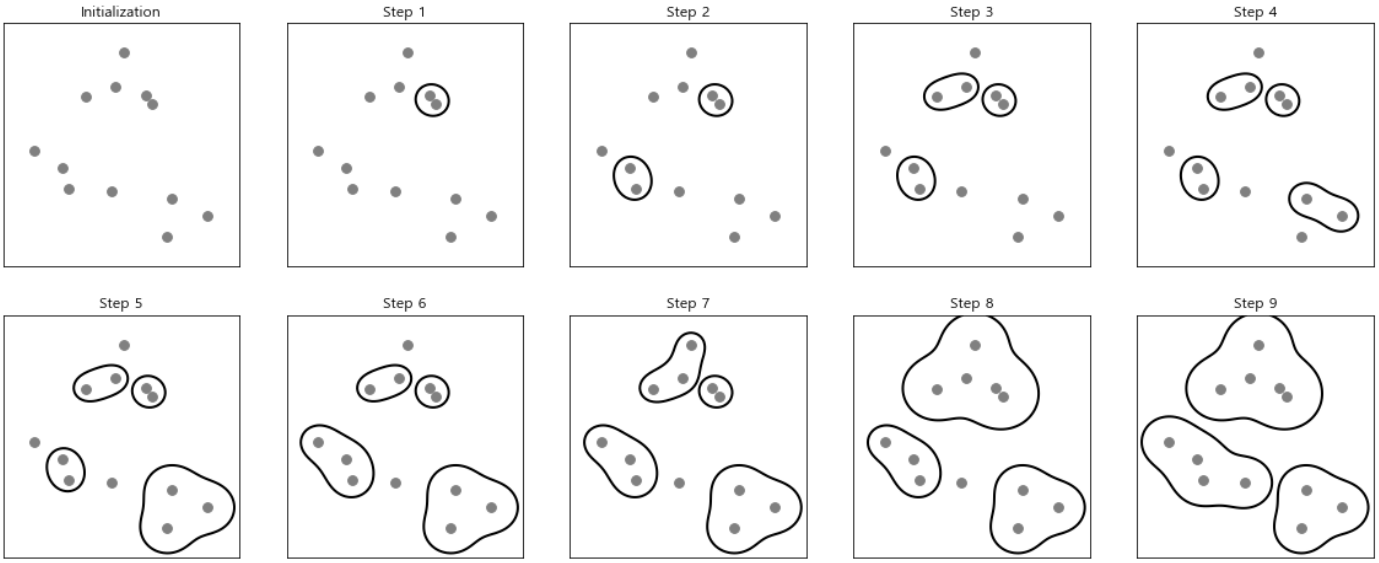
```
import mglearn
from sklearn.datasets import make_moons
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

#인위적인 2차원 데이터 셋 생성
X_varied, y_varied = make_moons(n_samples=200,noise=0.05, random_state=0)

# 3개의 클러스터 생성
y_pred = KMeans(n_clusters=3).fit_predict(X_varied)

mglearn.discrete_scatter(X_varied[:,0], X_varied[:,1], y_pred)
plt.legend(["cluster 0","cluster 1","cluster 2"],loc="best")
plt.xlabel("feature 0")
plt.ylabel("feature 1")
plt.show()
```



[군집 (Clustering)] - 병합 군집

chapter01_sklearn/ex18.py 병합 군집 과정 보기

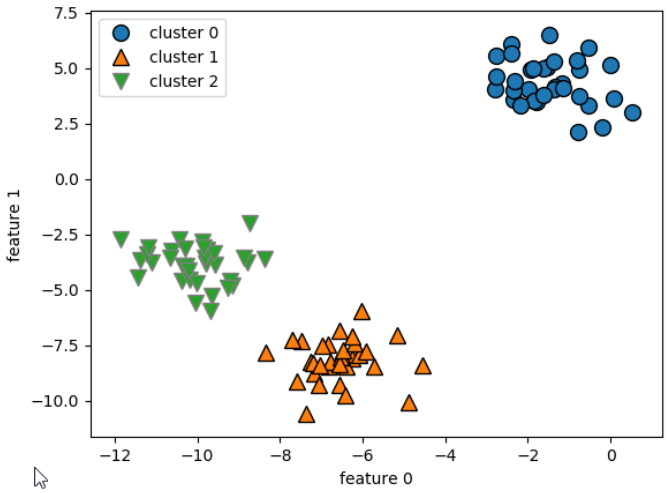
```
import mglearn
mglearn.plots.plot_agglomerative_algorithm()
```



초기에 각 포인트가 하나의 클러스터입니다. 그 다음 각 단계에서 가장 가까운 두 클러스터가 합쳐집니다. 목표 클러스터 개수까지 다다르면 합치는 것을 멈춥니다.

chapter01_sklearn/ex19_agglomerative.py

```
import mglearn
from sklearn.datasets import make_blobs
from sklearn.cluster import AgglomerativeClustering
import matplotlib.pyplot as plt

#인위적인 2차원 데이터 셋 생성
X , y = make_blobs( random_state=1)

# 3개의 클러스터 생성
agg= AgglomerativeClustering(n_clusters=3)
assignment = agg.fit_predict(X)

mglearn.discrete_scatter(X [:,0], X [:,1], assignment)
plt.legend(["cluster 0","cluster 1","cluster 2"],loc="best")
plt.xlabel("feature 0")
plt.ylabel("feature 1")
plt.show()
```

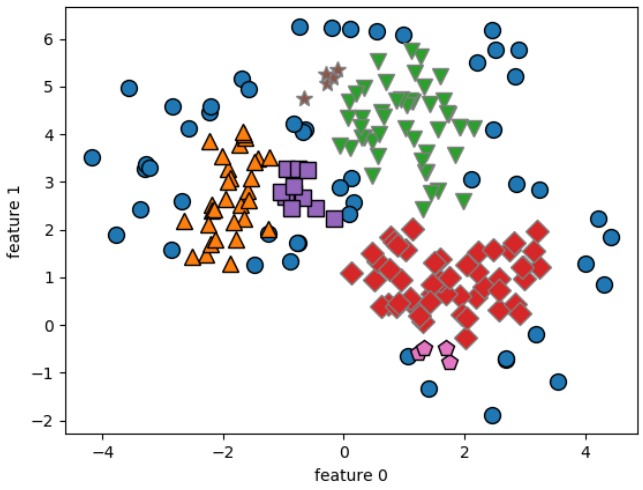


[군집 (Clustering)] - DBSCAN

주요 장점은 클러스터의 개수를 미리 지정할 필요가 없다는 점입니다.
어떤 복잡한 형상도 찾을 수 있으며, 어떤 클래스에도 속하지 않는 포인트를 구분할 수 있습니다.
병합 군집이나 K-Means보다는 다소 느리지만 비교적 큰 데이터셋에 적용할 수 있습니다.
밀집 지역이 한 클러스터를 구성하며 비교적 비어 있는 지역을 경계로 다른 클러스터와 구분됩니다.
시작할 때 무작위로 포인트를 선택합니다. 그런 다음 그 포인트에서 eps 거리 안에 min_samples 보다 많은 포인트가 있다면 그 포인트는 핵심 샘플로 레이블하고, 새로운 클러스터 레이블을 할당합니다. 그런 다음 그 포인트의 eps 안의 모든 이웃을 살핍니다. 클러스터는 eps 거리 안에 더 이상 핵심 샘플이 없을 때까지 자라납니다.
eps 또는 min_sample 값을 조정하여 클러스터 수 간접 조정이 가능합니다.

chapter01_sklearn/ex20_dbscan.py

```
import mglearn
from sklearn.datasets import make_blobs
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt

#인위적인 2차원 데이터 셋 생성
X , y  = make_blobs(n_samples=200, random_state=0)

dbscan = DBSCAN()

clusters = dbscan.fit_predict(X )

mglearn.discrete_scatter(X [:,0], X[:,1], clusters)
plt.xlabel("feature 0")
plt.ylabel("feature 1")
plt.show()
```



chapter01_sklearn/ex21_dbscan.py

```
import mglearn
from sklearn.datasets import make_moons
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt

#인위적인 2차원 데이터 셋 생성
X , y = make_moons(n_samples=200,noise=0.05, random_state=0)

dbscan = DBSCAN()
dbscan.eps = 0.2  # 간접적으로  클러스터링 수 조절
clusters = dbscan.fit_predict(X )

mglearn.discrete_scatter(X [:,0], X [:,1], clusters  )
plt.legend(["cluster 0","cluster 1","cluster 2"],loc="best")
plt.xlabel("feature 0")
plt.ylabel("feature 1")
plt.show()
```