군집 (clustering)

특성만 알고 있는 상황에서 훈련과 모델 평가에 필요한 타깃 데이터가 없기 때문에 지도 학습을 사용할 수 없는 경우 비지도 학습을 사용합니다.
군집(clustering) 알고리즘의 목적은 샘플에 잠재되어 있는 그룹을 식별하는 것입니다.

군집 (clustering)

- K-평균을 사용한 군집
  - 샘플을 k 개의 그룹으로 나눌때 k-평균(k-means) 군집을 사용합니다.
  - k-평균(k-means) 군집 알고리즘은 샘플을 k개의 그룹으로 나눕니다. 각 그룹은 거의 동일한 분산을 가집니다.
  - 그룹의 개수 k는 하이퍼파라미터로 사용자가 지정해야 합니다.

  - k-평균 동작 방식

1. k개의 클러스터 '중심' 포인트를 랜덤한 위치에 만듭니다.
2. 각 샘플에 대해
   a. 각 샘플과 k개의 중심 포인트 사이 거리를 계산합니다.
   b. 샘플을 가장 가까운 중심 포인트의 클러스터에 할당합니다.
3. 중심 포인트를 해당하는 클러스터의 평균(중심)으로 이동합니다.
4. 더 이상 샘플의 클러스터 소속이 바뀌지 않을 때까지 단계 2와 단계 3을 반복합니다

  - k-평균 군집은 클러스터가 둥그런 모양으로 간주합니다.
  - k-평균 군집은 모든 특성은 동일한 스케일을 가정합니다. (가정에 맞도록 특성을 표준화해야 합니다.)
  - k-평균 군집은 클러스터 크기는 균형 잡혀 있습니다.

군집 (clustering)

- K-평균을 사용한 군집
  - 사이킷런에는 KMeans 클래스에 k-평균 군집이 구현되어 있습니다
  - n_clusters 매개변수가 클러스터 k의 수를 지정합니다.
  - labels_ 속성에서 각 샘플의 예측 클래스를 확인할 수 있습니다.

```
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

iris = datasets.load_iris()    # 데이터  로드
features = iris.data

scaler = StandardScaler()  # 특성 표준화
features_std = scaler.fit_transform(features)

cluster = KMeans(n_clusters=3, random_state=0, n_jobs=-1)  # k-평균 객체 생성
model = cluster.fit(features_std)    # 모델  훈련
model.labels_           # 예측 클래스 확인
iris.target             # 진짜 클래스  확인

new_observation = [[0.8, 0.8, 0.8, 0.8]]   #New Sample Data
model.predict(new_observation)    # 샘플의 클러스터를 예측
model.cluster_centers_
```

군집 (clustering)

- K-평균을 사용한 군집
  - K-Means++ 알고리즘은 중심 포인트 하나를 먼저 랜덤하게 선택하고 그 다음부터는 이전 중심 포인트와의 거리를 고려하여 다음 중심 포인트를 선택합ㄴ디ㅏ.
  - 사이킷런의 KMeans 클래스의 init 매개변수 기본값은 k-means++입니다
  - n_init 횟수만큼 반복하여 최상의 결과를 만드는 중심 포인트를 찾습니다. (기본값 10)
  - 비교하는 기준은 샘플과 클러스터 중심까지의 거리 합(이너셔, inertia_속성에 저장)입니다. (score 메서드에서 반환하는 값)
  - 샘플 데이터를 각 클러스터까지 거리로 변환하는 transform()를 제공합니다.

```
model.inertia_
model.score(features_std)
model.transform(new_observation)

inertia = []
for n in range(1, 10):
    kmeans = KMeans(n_clusters=n, random_state=0, n_jobs=-1)
    inertia.append(kmeans.fit(features_std).inertia_)

import matplotlib.pyplot as plt
plt.plot(range(1, 10), inertia)
plt.show()
```

군집 (clustering)

- K-평균 군집 속도 향상
  - 미니배치 k-평균은 랜덤 샘플에 대해서만 수행합니다. (성능을 조금만 희생하고 알고리즘 학습에 드는 시간을 대폭 줄여줍니다.)
  - batch_size 매개변수는 각 배치에 랜덤하게 선택할 샘플의 수를 조절합니다
  - 훈련 세트가 너무 크면 하나의 넘파일 배열로 전달하기 어려우며, 데이터를 조금씩 전달하면서 훈련할 수 있는 partial_fit()를 사용합니다.
  - 실전에서는 파일 같은 외부 저장소에서 시스템의 메모리가 허용하는만큼 데이터를 추출하여 모델을 훈련합니다

```
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import MiniBatchKMeans

iris = datasets.load_iris()  # 데이터 로드
features = iris.data
scaler = StandardScaler()  # 특성을 표준화
features_std = scaler.fit_transform(features)

cluster = MiniBatchKMeans(n_clusters=3, random_state=0, batch_size=100)
model = cluster.fit(features_std)  # 모델 훈련

mb_kmeans = MiniBatchKMeans()
for i in range(3):
    mb_kmeans.partial_fit(features_std[i*50:(i+1)*50])
```

군집 (clustering)

- 평균이동을 사용한 군집
  - 사이킷런의 평균 이동 구현인 MeanShift는 클러스터 수나 모양을 가정하지 않고 샘플을 그룹으로 나눌때 사용합니다
  - bandwidth는 샘플이 이동 방향을 결정하기 위해 사용하는 면적(커널)의 반경을 지정합니다.

```
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import MeanShift

iris = datasets.load_iris()  # 데이터 로드
features = iris.data

scaler = StandardScaler()  # 특성을 표준화
features_std = scaler.fit_transform(features)
cluster = MeanShift(n_jobs=-1)  # meanshift 객체 생성
model = cluster.fit(features_std)  # 모델 훈련

model.labels_
model.cluster_centers_
```

군집 (clustering)

- DBSCAN을 사용한 군집
  - DBSCAN은 샘플의 밀집 영역을 클러스터로 그룹핑할 수 있습니다.
  - eps매개변수 : 다른 샘플을 이웃으로 고려하기 위한 최대 거리
  - min_samples : 핵심 샘플로 간주하기 위해 eps 거리 내에 필요한 최소 샘플 개수
  - metric : eps에서 사용할 거리 측정 방식
  - DBSCAN에서 찾은 핵심 샘플의 인덱스는 core_sample_indices_속성에 저장되어 있습니다.
  - 훈련 데이터에 대한 예측 결과를 얻으려면 fit_predict()를 사용합니다.

```
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN

iris = datasets.load_iris()  # 데이터 로드
features = iris.data
scaler = StandardScaler()  # 특성 표준화
features_std = scaler.fit_transform(features)

cluster = DBSCAN(n_jobs=-1)      # DBSCAN 객체 생성
model = cluster.fit(features_std)  # 모델 훈련
model.labels_        # 클러스터 소속을 확인
model.core_sample_indices_
cluster.fit_predict(features_std)
```

군집 (clustering)

- 계층적 병합을 사용한 군집
  - 병합 군집(agglometrative clustering)은 강력하고 유연한 계층적 군집 알고리즘입니다.
  - 병합 군집은 모든 샘플이 각자 하나의 클러스터로 시작합니다. 그 다음 어떤 조건에 부합하는 클러스터들이 서로 병합됩니다.
  - 이 과정이 어떤 종료 조건에 도달할 때까지 반복되어 클러스터가 커집니다.
  - 사이킷런의 AgglomerativeClustering 클래스는 linkage 매개변수를 사용하여 병합된 클러스터의 분산(ward) 또는 두 클러스터 샘플 간의 평균 거리(average) 또는 두 클러스터 샘플 간의 최대 거리(complete)를 최소화하는 병합 전략을 결정합니다.
  - affinity매개변수는 linkage에서 사용할 거리 측정 방식을 결정합니다. (minkowski, euclidean등)
  - n_clusters는 군집 알고리즘이 찾을 클러스터 수를 지정합니다.
  - n_clusters개의 클러스터가 남을 때까지 연속적으로 병합됩니다
  - labels_속성을 사용해 각 샘플이 속한 클러스터를 확인할 수 있습니다.
  - linkage 매개변수에 두 클러스터 샘플 간의 최소 거리를 최소화하는 병합 전략인 single 옵션이 추가되었습니다.

군집 (clustering)

- 계층적 병합을 사용한 군집

```
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering

iris = datasets.load_iris()   # 데이터 로드
features = iris.data

scaler = StandardScaler()   # 특성  표준화
features_std = scaler.fit_transform(features)

cluster = AgglomerativeClustering(n_clusters=3)   # 병합 군집 객체 생성
model = cluster.fit(features_std)    # 모델  훈련
model.labels_            # 클러스터 소속을 확인
cluster.fit_predict(features_std)
```