

1 데이터분석 단계(Data Analysis Cycle)

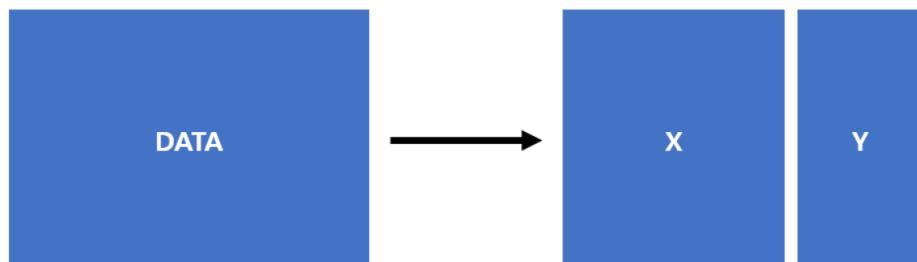
[Open in Colab](#)

✓ 데이터 전처리: (0) 쓸모 없을 뻔한 Raw를 쓸모 있는 Data로 변환

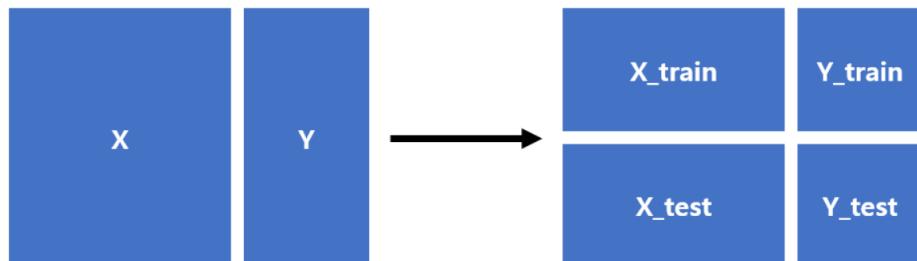
100	T50	횟수	111	TPU	...
few	Gds	Hvi	Rew	Fa	...
Fre	CT	QTP	D	합	...
'1'	1	23	22	NaN	43
76	NaN	43	32	1	8
'Hi'	NaN	NaN	NaN	NaN	87
23	98	NaN	64	46	NaN
c	90	'WW'	24	'KK'	4
t	NaN	2	NaN	NaN	6
64	NaN	90	'IU'	4	76

번호	시간	총량	기간	누적	...
1	1	23	22	21	43
76	33.3	43	32	1	8
5	33.3	52	35	21	87
23	98	52	64	46	61
90	33.3	2	24	33	4
55	2	52	35	21	6
64	33.3	90	11	4	76

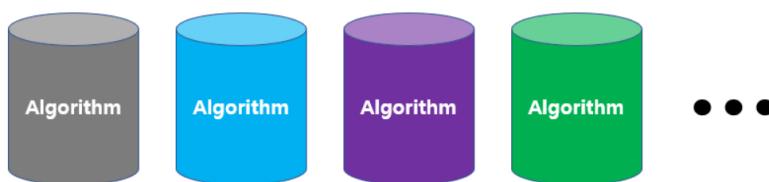
✓ 데이터 분할: (1) 목표/종속변수 Y와 설명/독립변수 X설정



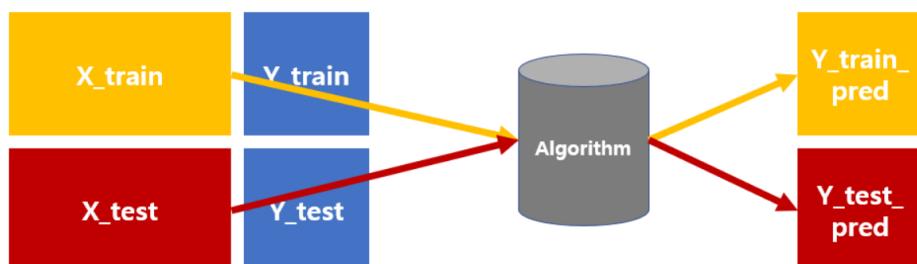
✓ 데이터 분할: (2) 학습데이터 Train과 예측 데이터 Test로 분할



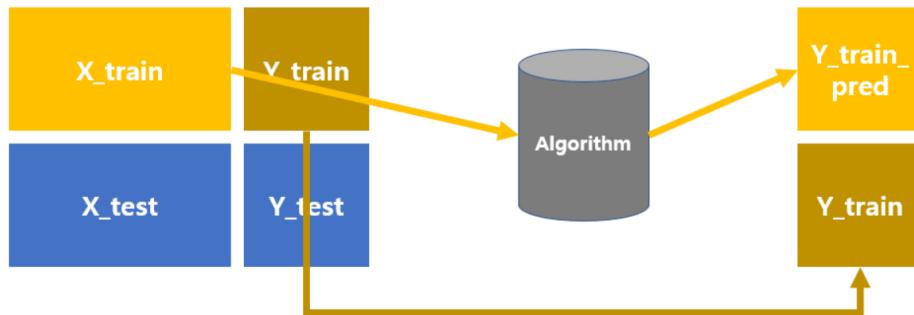
✓ 모델링: (3) 분석 목적에 맞는 알고리즘(Base & Advanced) 후보들 준비



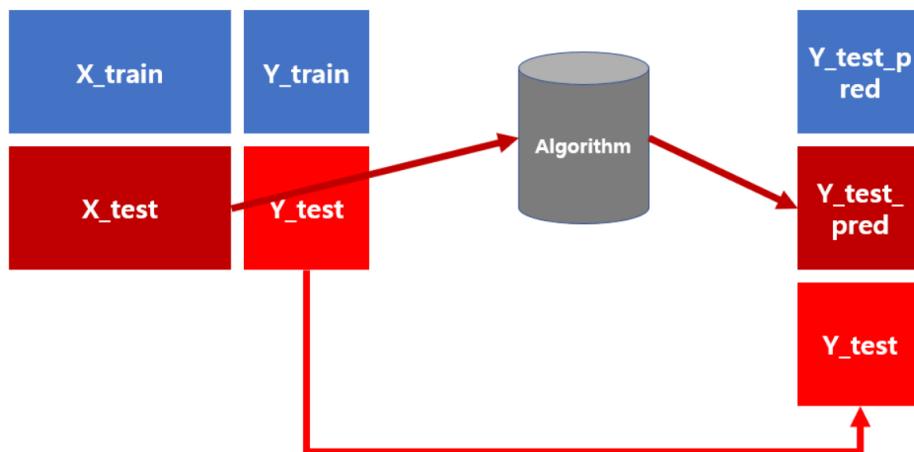
✓ 모델링 & 학습: (4) 알고리즘 평가를 위해 Train/Test의 예측값 추정



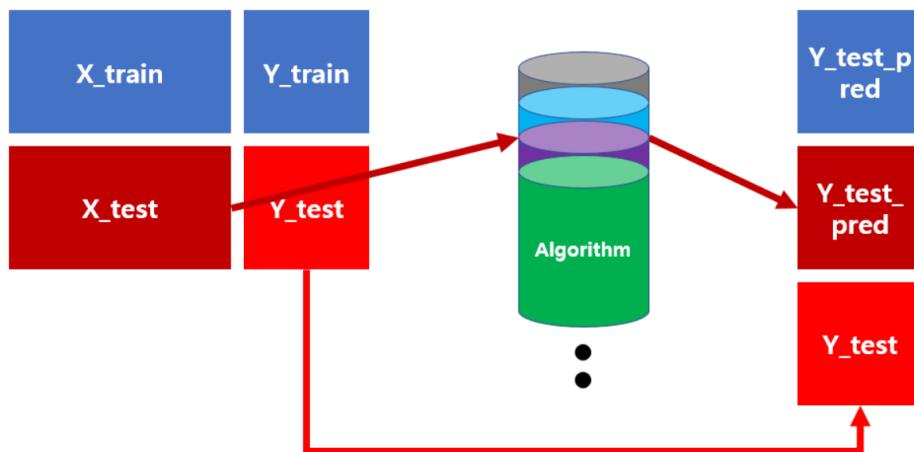
✓ 평가: (5) 학습(Train)이 잘 되었는지 알고리즘 성능검증



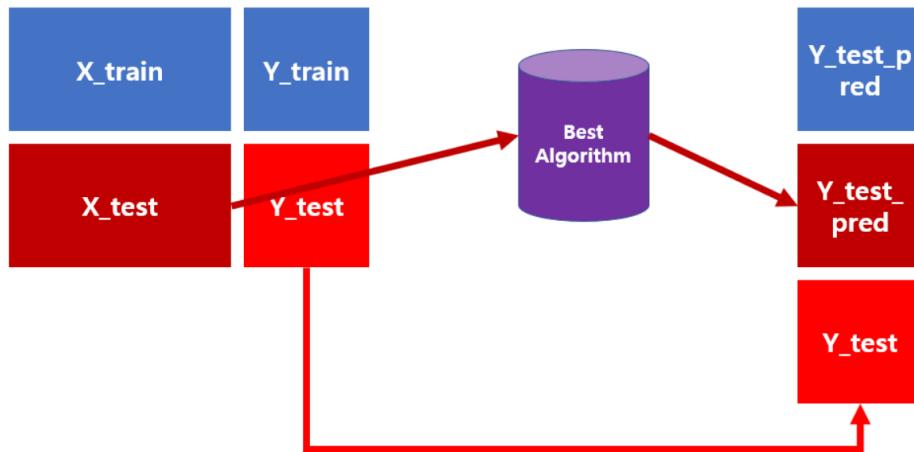
✓ 평가: (6) 예측(Test)이 잘 되었는지 알고리즘 성능검증



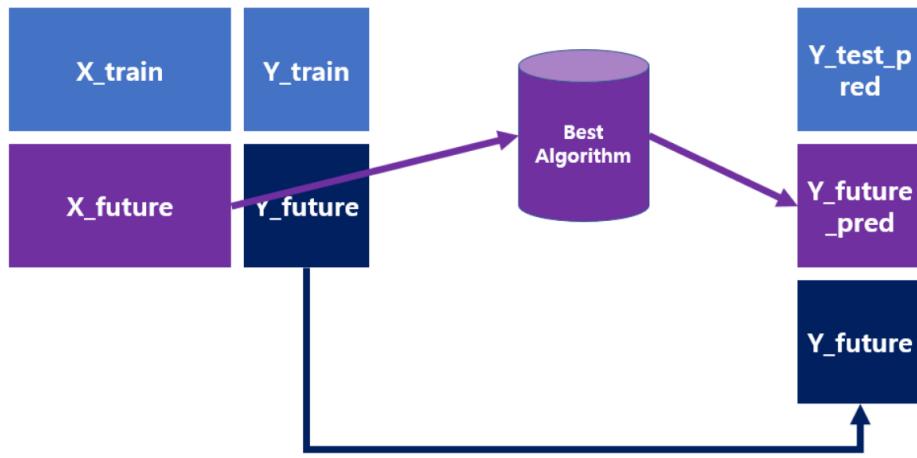
✓ 최적 알고리즘 선택: (7) 알고리즘을 변경하여 위 과정 반복 후



✓ 최적 알고리즘 선택: (7) 최고 성능의 알고리즘 선택

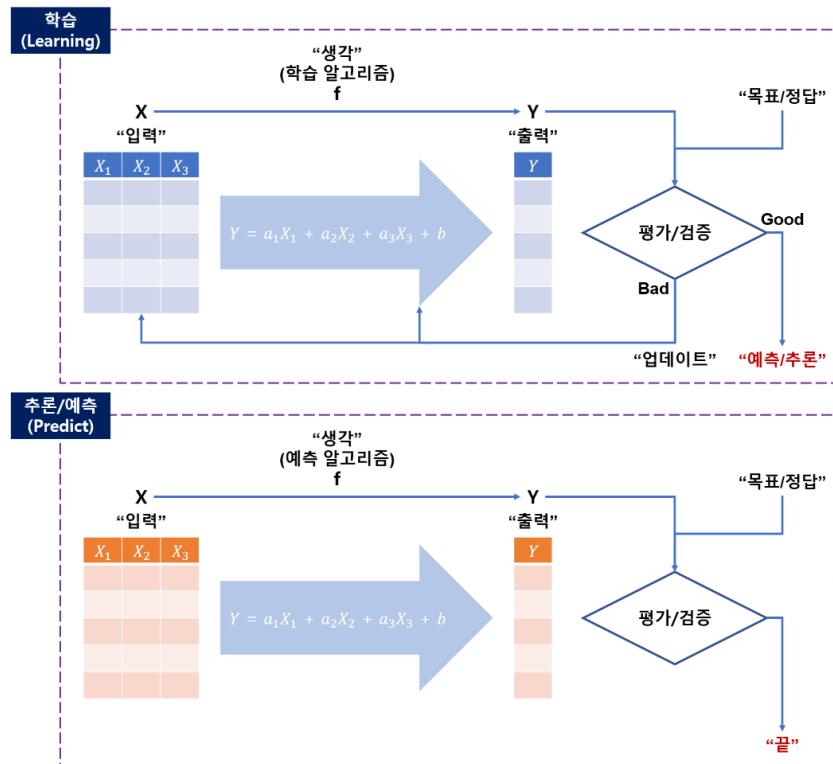


✓ 배포: (8) 실제 비즈니스 서비스 현업 적용 및 매출/수익/개선 정도 평가



2 비지도학습(Unsupervised) 알고리즘: 군집분석

- 데이터분석 과정: 학습 + 추론/예측



“비지도학습(Unsupervised Learning)은 정답 레이블이 없기 때문에, 주로 데이터를 새롭게 표현하여 원래 데이터보다 쉽게 해석하거나 특성을 추가적으로 파악하는데 주로 사용”

- 군집분석: 비지도학습 알고리즘 중 군집화를 위해 사용되는 가장 기본(Baseline) 알고리즘

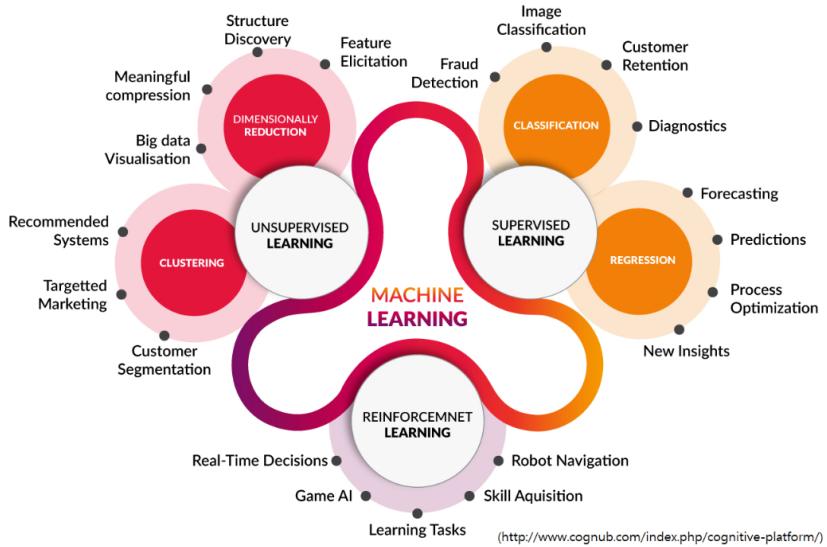
(비수학적) “일상 속 문제들은 어떤 유형들이 있는지 파악하는 문제”

- 고객들의 정보를 통해 성인인지 미성년자인지 정답을 찾는 문제가 분류문제
- 고객들의 정보를 통해 어떤 쇼핑 취향들이 있는지 성인 또는 미성년자 레이블을 할당하며 추론하는 문제가 군집문제

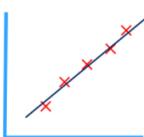
- 데이터가 2차원 일 경우 시각화를 통해 눈으로도 패턴, 군집, 관계를 어림짐작 가능
- 데이터가 3차원 이상 일 경우 시각화로 패턴, 군집, 관계 파악 어려움

(수학적) “특정 출력(종속변수)/입력(독립변수)의 구분이나 관계 추론도 없고 학습을 위한 목표값도 없이, 주어진 데이터를 유사한 그룹으로 군집화(Clustering) 하는 알고리즘”

- 분류문제: 데이터 변수(Feature, Variable)들을 사용하여 특정 분류값을 예측
- 군집문제: 데이터 변수(Feature, Variable)들을 사용하여 여러개의 레이블을 할당하면서 특정 군집값(Cluster)을 예측

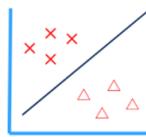


- How much is the stock of Samsung Electronics tomorrow?



Regression – Looking for a statistical relationship across variables that may give us an estimate of a particular outcome. (Supervised)

- Will Samsung Electronics' stocks rise or fall tomorrow?



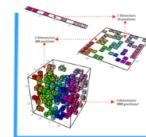
Classification – Similar to regression but looking for separations in the data given predefined classes. (Supervised)

- Are Samsung Electronics and Naver similar business companies?



Clustering – Do not have predefined classes but trying to find groups or sets based upon data at hand. (Unsupervised)

- What are the representatives among all stocks in the KOSPI?



Dimensionality Reduction – Transformation of data from high-dimensional into a low-dimensional space so that it retains some meaningful properties of the origin data. (Unsupervised)

Clustering Algorithms	Association Rule Learning Algorithms	Dimensionality Reduction Algorithms	Ensemble Algorithms	Deep Learning Algorithms
K-Means	Apriori algorithm	Principal Component Analysis (PCA)	Boosting	Deep Boltzmann Machine (DBM)
K-Medians	Eclat algorithm	Principal Component Regression (PCR)	Bootstrapped Aggregation (Bagging)	Deep Belief Networks (DBN)
Expectation Maximisation (EM)	(A,B) → C (D,E) → F (A,E) → G	-	AdaBoost	Convolutional Neural Network (CNN)
Hierarchical Clustering	-	Partial Least Squares Regression (PLSR)	Stacked Generalization (blending)	Stacked Auto-Encoders
-	-	Sammon Mapping	Gradient Boosting Machines (GBM)	-
-	-	Multidimensional Scaling (MDS)	Gradient Boosted Regression Trees (GBRT)	-
-	-	Projection Pursuit	Random Forest	-
-	-	-	-	-
-	-	Linear Discriminant Analysis (LDA)	-	-
-	-	Mixture Discriminant Analysis (MDA)	-	-
-	-	Quadratic Discriminant Analysis (QDA)	-	-
-	-	Flexible Discriminant Analysis (FDA)	-	-

- 종류: 군집문제 해결 알고리즘은 다양하고, 데이터 특성/구조/목적에 맞는 적절한 선택 필요

- **Hard Clustering:** 데이터 각 샘플(Row)이 여러개의 군집들 중 하나의 군집에만 포함되는 것만 허용하는 방식 (One Sample ∈ One Cluster)

- Partitional Clustering

- **Soft Clustering:** 데이터 각 샘플(Row)이 여러개의 군집들 중 중복을 반영하여 여러 군집에 포함되는 것을 허용하는 방식으로, 가중치(Weight)나 확률(Probability)으로 군집에 속할 가능성으로 표현 (One Sample ∈ Multiple Cluster)

- hierarchical Clustering
- Density-based Clustering
- Grid-based Clustering
- Model-based Clustering

군집특성 분류	접근방법	측정기준	알고리즘	설명
Hard Clustering	Partitional Clustering	-	-	전체 데이터를 특정 기준에 의해 한번에 구분하는 방식
		Distance-based	K-Means K-Median K-Medoid Fuzzy Clustering PAM(Partitioning Around Medoids) CLARA(Clustering LARge Applications) CLARANS(Clustering Large Applications based on Randomized Search)	
Soft Clustering	Hierarchical Clustering	-	-	가까운 데이터들을 차근차근 끌어나가는 방식
		Agglomerative (Bottom-up)	Single Linkage(Graph-based) Complete Linkage(Graph-based) Average Linkage(Graph-based) Centroid Linkage(Distance-based) Ward Linkage(Distance-based) AGNES(AGglomeration NESting)	개별 데이터에서 유사한 데이터끼리 끌어가는 방식 각 군집에서 1개씩 뽑았을 때 나타날 수 있는 최솟값으로 군집간 거리 측정 각 군집에서 1개씩 뽑았을 때 나타날 수 있는 최댓값으로 군집간 거리 측정 모든 데이터에 대한 군집간 거리평균 측정 2개의 군집간 중심사이의 거리 측정 군집 내 오차 제곱합 기반으로 측정(유일하게 군집내 거리 기반) 모든 데이터를 하나의 군집이라 가정 후 세부 군집으로 분리하는 방식
		Divisive (Top-down)	DIANA(Divisive ANAlysis) BIRCH(Balanced Iterative Reducint and Clustering Using Hierarchies) CURE(Clustering Using Representatives) Chameleon	AGNES의 역순으로 측정 Clustering Feature Tree 기반으로 양질의 데이터에 좋은 성능 이상치에 대해서 굉장히 Robust하게 설계 CURE & DBSCAN 보다 좋은 성능이며 임의 형태도 잘 찾으나 컴퓨팅 느낌
	Density-based Clustering	-	DBSCAN(Density Based Spatial Clustering of Applications with Noise) OPTICS(Ordering Points To Identify the Clustering Structure) DENCLUE(DENsity-based CLUstErIng) Density-peaks Robust-DB(Density Based)	거리 기반 군집화는 임의 형태 군집 추정에 어려우며 형태 반영을 위한 방식 Kmeans는 거리기준 구분, DBSCAN은 데이터밀도 높은 부분 군집화 방식
	Grid-based Clustering	-	STING(Statistical Information Grid) WaveCluster CLIQUE(CLustering In QUEst)	유한개의 공간격자로 매핑하여 빠른 처리 가능한 방식
	Model-based Clustering	-	Gaussian Mixture Algorithm Expectation Maximization Algorithm AutoClass(Mixture of Naive Bayes) Cobweb	군집에 모델을 가정하기에 통계적 또는 인공 신경망적 접근 방식
		Network-based	Kohonen Clustering SOM(Self-Organizing Map)	

• Target Algorithm:

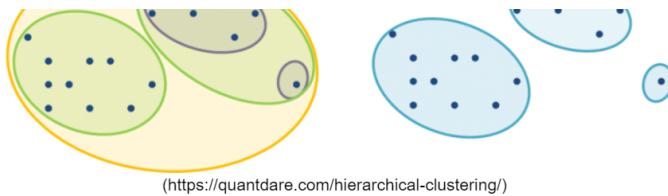
(1) Partitional Clustering vs Hierarchical Clustering

- **Partitional:** 전체 데이터를 Hard Clustering 기준으로 한번에 군집형성 하는 방식
- **Hierarchical:** 각각의 데이터에서 유사성 척도(Similarity Measure)에 의해 가까운 데이터들을 Tree 형태의 계층적 군집으로 차근 차근 끌어나가는 방식이며 Tree에서 어느 수준을 기준으로 하느냐에 따라 군집이 달라짐

Hierarchical Clustering

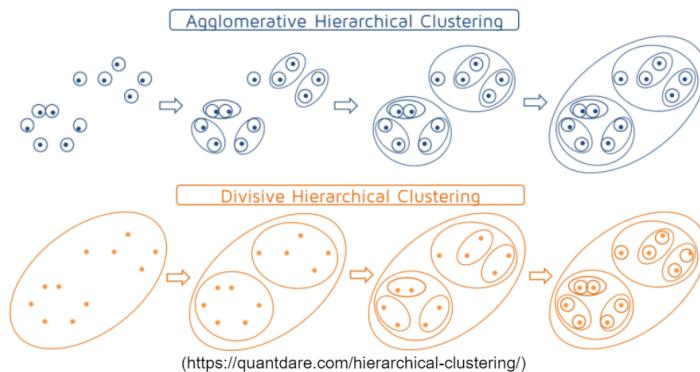
Partitional Clustering





(2) Hierarchical Clustering

- **Agglomerative(Bottom-up):** 개별 데이터에서 유사한 데이터끼리 묶어가는 방식
- **Divisive(Top-down):** 모든 데이터를 하나의 군집이라 가정 후 세부 군집으로 분리하는 방식



군집특성 분류	접근방법	측정기준	알고리즘
Hard Clustering	Partitional Clustering	Distance-based	K-Means K-Median K-Mode K-Medoid Fuzzy Clustering PAM(Partitioning Around Medoids) CLARA(Clustering LARge Applications) CLARANS(Clustering Large Applications based on RANDomized Search)
Soft Clustering	Hierarchical Clustering	Agglomerative (Bottom-up)	Single Linkage(Graph-based) Complete Linkage(Graph-based) Average Linkage(Graph-based) Centroid Linkage(Distance-based) Ward Linkage(Distance-based) AGNES(AGglomerative NESting) DIANA(DiVisive ANAlysis)
		Divisive (Top-down)	BIRCH(Balanced Iterative Reducint and Clustering Using Hierarchies) CURE(Clustering Using Representatives) Chameleon
	Density-based Clustering		DBSCAN(Density Based Spatial Clustering of Applications with Noise) OPTICS(Ordering Points To Identify the Clustering Structure) DENCLUE(DENsity-based CLUstErting) Density-peaks Robust-DB(Density Based)
	Grid-based Clustering		STING(Statistical Information Grid) WaveCluster CLIQUE(CLustering In QUEst)
	Model-based Clustering	Distribution-based	Gaussian Mixture Algorithm Expectation Maximization Algorithm AutoClass(Mixture of Naïve Bayes) Cobweb
		Network-based	Kohonen Clustering SOM(Self-Organizing Map)

3 예제 데이터셋(Dataset)

3.1 statsmodels 모듈 사용 데이터셋

```
# 라이브러리 불러오기
import statsmodels.api as sm
```

- 대기중 CO2 농도 데이터:

```

data = sm.datasets.get_rdataset("CO2", package="datasets")

• 황체형성 호르몬(Luteinizing Hormone)의 수치 데이터:
data = sm.datasets.get_rdataset("lh")

• 1974~1979년 사이의 영국의 호흡기 질환 사망자 수 데이터:
data = sm.datasets.get_rdataset("deaths", "MASS")

• 1949~1960년 사이의 국제 항공 운송인원 데이터:
data = sm.datasets.get_rdataset("AirPassengers")

• 미국의 강수량 데이터:
data = sm.datasets.get_rdataset("precip")

• 타이타닉호의 탑승자들에 대한 데이터:
data = sm.datasets.get_rdataset("Titanic", package="datasets")

```

- **data**가 포함하는 정보:

- `package`: 데이터를 제공하는 R 패키지 이름
- `title`: 데이터 이름
- `data`: 데이터를 담고 있는 데이터프레임
- `__doc__`: 데이터에 대한 설명 문자열(R 패키지의 내용 기준)

3.2 sklearn 모듈 사용 데이터셋

1) 패키지에 포함된 데이터(`load` 명령어)

```

# 라이브러리 불러오기
from sklearn.datasets import load_boston

• load_boston: 회귀용 보스턴 집값
raw = load_boston()
print(raw.DESCR)
print(raw.keys())
print(raw.data.shape, raw.target.shape)

• load_diabetes: 회귀용 당뇨병 자료
• load_linnerud: 회귀용 linnerud 자료
• load_iris: 분류용 붓꽃(iris) 자료
• load_digits: 분류용 숫자(digit) 필기 이미지 자료
• load_wine: 분류용 포도주(wine) 등급 자료
• load_breast_cancer: 분류용 유방암(breast cancer) 진단 자료

```

2) 인터넷에서 다운로드 할 수 있는 데이터(`fetch` 명령어)

```

# 라이브러리 불러오기
from sklearn.datasets import fetch_california_housing

• fetch_california_housing: 회귀용 캘리포니아 집값
raw = fetch_california_housing()
print(raw.DESCR)
print(raw.keys())
print(raw.data.shape, raw.target.shape)

• fetch_covtype: 회귀용 토지 조사 자료
• fetch_20newsgroups: 뉴스 그룹 텍스트 자료
• fetch_olivetti_faces: 얼굴 이미지 자료
• fetch_lfw_people: 유명인 얼굴 이미지 자료
• fetch_lfw_pairs: 유명인 얼굴 이미지 자료
• fetch_rcv1: 로이터 뉴스 말뭉치
• fetch_kddcup99: Kddcup 99 Tcp dump 자료

```

3) 확률분포를 사용한 가상 데이터(`make` 명령어)

```

# 라이브러리 불러오기
from sklearn.datasets import make_regression

• make_regression: 회귀용 가상 데이터
X, y, c = make_regression(n_samples=100, n_features=10, n_targets=1, bias=0, noise=0, coef=True, random_state=0)

• make_classification: 분류용 가상 데이터 생성
• make_blobs: 클러스터링용 가상 데이터 생성

```

4) `load/fetch` 명령어 데이터에서 `raw`가 포함하는 정보: Bunch라는 클래스 객체 형식으로 생성

- data : (필수) 독립 변수 ndarray 배열
- target : (필수) 종속 변수 ndarray 배열
- feature_names : (옵션) 독립 변수 이름 리스트
- target_names : (옵션) 종속 변수 이름 리스트
- DESCR : (옵션) 자료에 대한 설명

3.3 군집문제 예시

```
In [4]: # Clustering
from sklearn.datasets import make_moons
X, y = make_moons(n_samples=1000, noise=0.05, random_state=123)
print(make_moons.__doc__)
print(X.shape, y.shape)

import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
data = pd.concat([pd.DataFrame(X), pd.DataFrame(y, columns=['label'])], axis=1)
plt.figure(figsize=(10,10))
sns.pairplot(data=data.iloc[:,2:], kind='scatter')
# sns.pairplot(data=data, kind='scatter', hue='label')
plt.show()
executed in 602ms, finished 02:02:15 2022-04-23
```

Make two interleaving half circles.

A simple toy dataset to visualize clustering and classification algorithms. Read more in the :ref:`User Guide <sample_generators>`.

Parameters

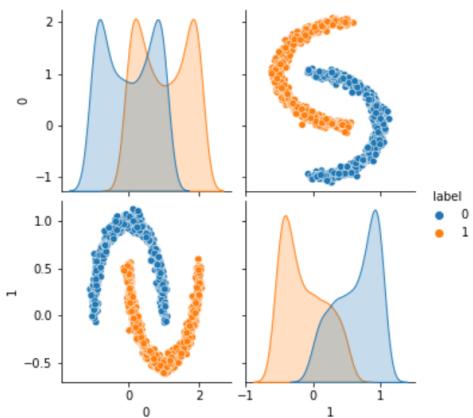
n_samples : int or tuple of shape (2,), dtype=int, default=100
 If int, the total number of points generated.
 If two-element tuple, number of points in each of two moons.
 ... versionchanged:: 0.23
 Added two-element tuple.
 shuffle : bool, default=True
 Whether to shuffle the samples.
 noise : float, default=None
 Standard deviation of Gaussian noise added to the data.
 random_state : int, RandomState instance or None, default=None
 Determines random number generation for dataset shuffling and noise.
 Pass an int for reproducible output across multiple function calls.
 See :term:`Glossary <random_state>`.

Returns

X : ndarray of shape (n_samples, 2)
 The generated samples.
 y : ndarray of shape (n_samples,)
 The integer labels (0 or 1) for class membership of each sample.

(1000, 2) (1000,)

<Figure size 720x720 with 0 Axes>



```
In [5]: # Clustering
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=100, random_state=123)
print(make_blobs.__doc__)
print(X.shape, y.shape)

import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
data = pd.concat([pd.DataFrame(X), pd.DataFrame(y, columns=['label'])], axis=1)
```

```

plt.figure(figsize=(10,10))
sns.pairplot(data=data.iloc[:,2:], kind='scatter')
# sns.pairplot(data=data, kind='scatter', hue='label')
plt.show()
executed in 685ms, finished 02:02:23 2022-04-23

```

Generate isotropic Gaussian blobs for clustering.

Read more in the :ref:`User Guide <sample_generators>`.

Parameters

- n_samples : int or array-like, default=100
 - If int, it is the total number of points equally divided among clusters.
 - If array-like, each element of the sequence indicates the number of samples per cluster.
- .. versionchanged:: v0.20
 - one can now pass an array-like to the ``n_samples`` parameter
- n_features : int, default=2
 - The number of features for each sample.
- centers : int or ndarray of shape (n_centers, n_features), default=None
 - The number of centers to generate, or the fixed center locations.
 - If n_samples is an int and centers is None, 3 centers are generated.
 - If n_samples is array-like, centers must be either None or an array of length equal to the length of n_samples.
- cluster_std : float or array-like of float, default=1.0
 - The standard deviation of the clusters.
- center_box : tuple of float (min, max), default=(-10.0, 10.0)
 - The bounding box for each cluster center when centers are generated at random.
- shuffle : bool, default=True
 - Shuffle the samples.
- random_state : int, RandomState instance or None, default=None
 - Determines random number generation for dataset creation. Pass an int for reproducible output across multiple function calls.
 - See :term:`Glossary <random_state>`.
- return_centers : bool, default=False
 - If True, then return the centers of each cluster.
- .. versionadded:: 0.23

Returns

- X : ndarray of shape (n_samples, n_features)
 - The generated samples.
- y : ndarray of shape (n_samples,)
 - The integer labels for cluster membership of each sample.
- centers : ndarray of shape (n_centers, n_features)
 - The centers of each cluster. Only returned if ``return_centers=True``.

See Also

- make_classification : A more intricate variant.

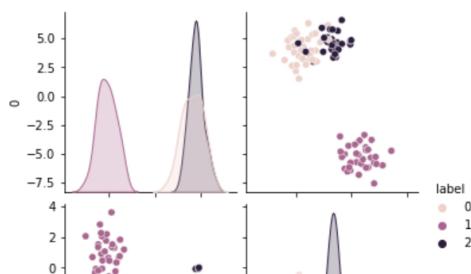
Examples

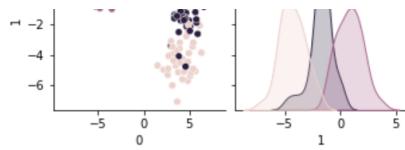
```

>>> from sklearn.datasets import make_blobs
>>> X, y = make_blobs(n_samples=10, centers=3, n_features=2,
...                     random_state=0)
>>> print(X.shape)
(10, 2)
>>> y
array([0, 0, 1, 0, 2, 2, 2, 1, 1, 0])
>>> X, y = make_blobs(n_samples=[3, 3, 4], centers=None, n_features=2,
...                     random_state=0)
>>> print(X.shape)
(10, 2)
>>> y
array([0, 1, 2, 0, 2, 2, 2, 1, 1, 0])
(100, 2) (100,)

<Figure size 720x720 with 0 Axes>

```





```
In [6]: # Clustering
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=1000,
                   n_features=3, cluster_std=[1.0, 2.5, 0.5], random_state=123)
print(make_blobs.__doc__)
print(X.shape, y.shape)

import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
data = pd.concat([pd.DataFrame(X), pd.DataFrame(y, columns=['label'])], axis=1)
plt.figure(figsize=(10,10))
sns.pairplot(data=data.iloc[:, :3], kind='scatter')
# sns.pairplot(data=data, kind='scatter', hue='label')
plt.show()
executed in 1.26s, finished 02:02:16 2022-04-23
```

Generate isotropic Gaussian blobs for clustering.

Read more in the :ref:`User Guide <sample_generators>`.

Parameters

`n_samples` : int or array-like, default=100
 If int, it is the total number of points equally divided among clusters.
 If array-like, each element of the sequence indicates the number of samples per cluster.

... versionchanged:: v0.20
 one can now pass an array-like to the ```n_samples``` parameter

`n_features` : int, default=2
 The number of features for each sample.

`centers` : int or ndarray of shape (`n_centers`, `n_features`), default=None
 The number of centers to generate, or the fixed center locations.
 If `n_samples` is an int and `centers` is None, 3 centers are generated.
 If `n_samples` is array-like, `centers` must be either None or an array of length equal to the length of `n_samples`.

`cluster_std` : float or array-like of float, default=1.0
 The standard deviation of the clusters.

`center_box` : tuple of float (min, max), default=(-10.0, 10.0)
 The bounding box for each cluster center when centers are generated at random.

`shuffle` : bool, default=True
 Shuffle the samples.

`random_state` : int, RandomState instance or None, default=None
 Determines random number generation for dataset creation. Pass an int for reproducible output across multiple function calls.
 See :term:`Glossary <random_state>`.

`return_centers` : bool, default=False
 If True, then return the centers of each cluster.

... versionadded:: 0.23

Returns

`X` : ndarray of shape (`n_samples`, `n_features`)
 The generated samples.

`y` : ndarray of shape (`n_samples`,)
 The integer labels for cluster membership of each sample.

`centers` : ndarray of shape (`n_centers`, `n_features`)
 The centers of each cluster. Only returned if ```return_centers=True```.

See Also

`make_classification` : A more intricate variant.

Examples

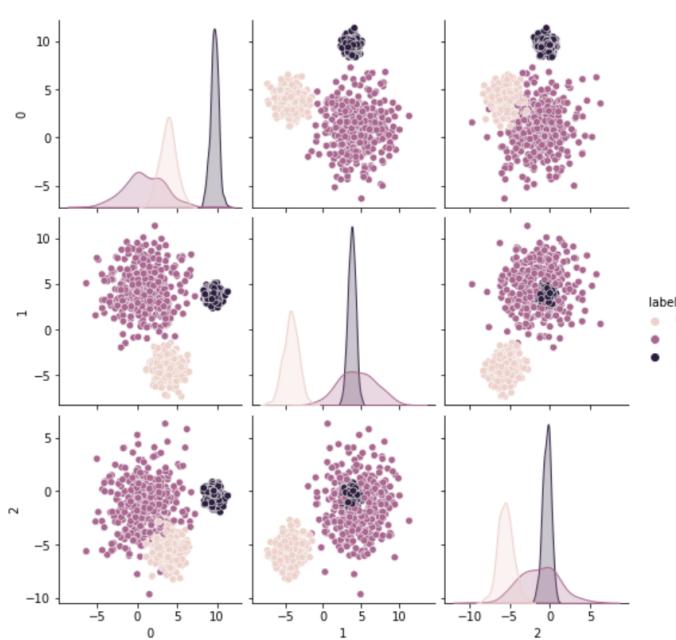
```
>>> from sklearn.datasets import make_blobs
>>> X, y = make_blobs(n_samples=10, centers=3, n_features=2,
...                     random_state=0)
>>> print(X.shape)
(10, 2)
>>> y
array([0, 0, 1, 0, 2, 2, 2, 1, 1, 0])
>>> X, y = make_blobs(n_samples=[3, 3, 4], centers=None, n_features=2,
...                     random_state=0)
>>> print(X.shape)
(10, 2)
>>> ..
```

```

>>> y
array([0, 1, 2, 0, 2, 2, 2, 1, 1, 0])
(1000, 3) (1000,)

<Figure size 720x720 with 0 Axes>

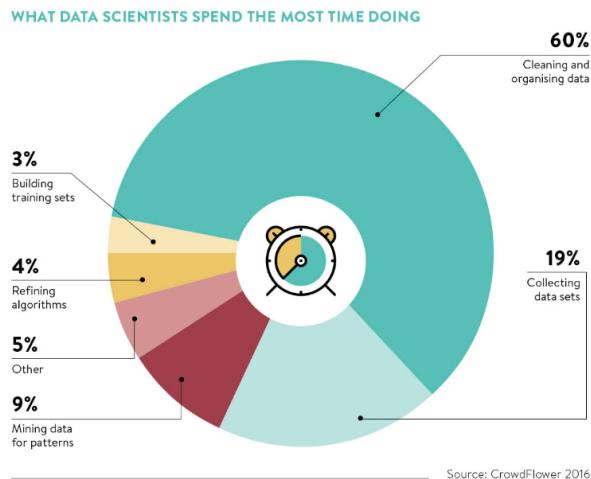
```



4 전처리 방향(Preprocessing)

- 목표:

- 대량으로 수집된 데이터는 그대로 활용 어려움
- 잘못 수집/처리 된 데이터는 엉뚱한 결과를 발생
- 알고리즘이 학습이 가능한 형태로 데이터를 정리



일반적인 전처리 필요항목:

- 데이터 결합
- 결측값 처리
- 이상치 처리
- 자료형 변환
- 데이터 분리
- 데이터 변환
- 스케일 조정 : 단위가 큰 변수들에 의해 군집화가 휘둘릴 수 있음!

5 함수세팅 및 추정 방향(Modeling): K-centroid Clustering

"K는 군집의 수를 의미하는 파라미터(Parameter)로 분석가가 미리 정해야 함."

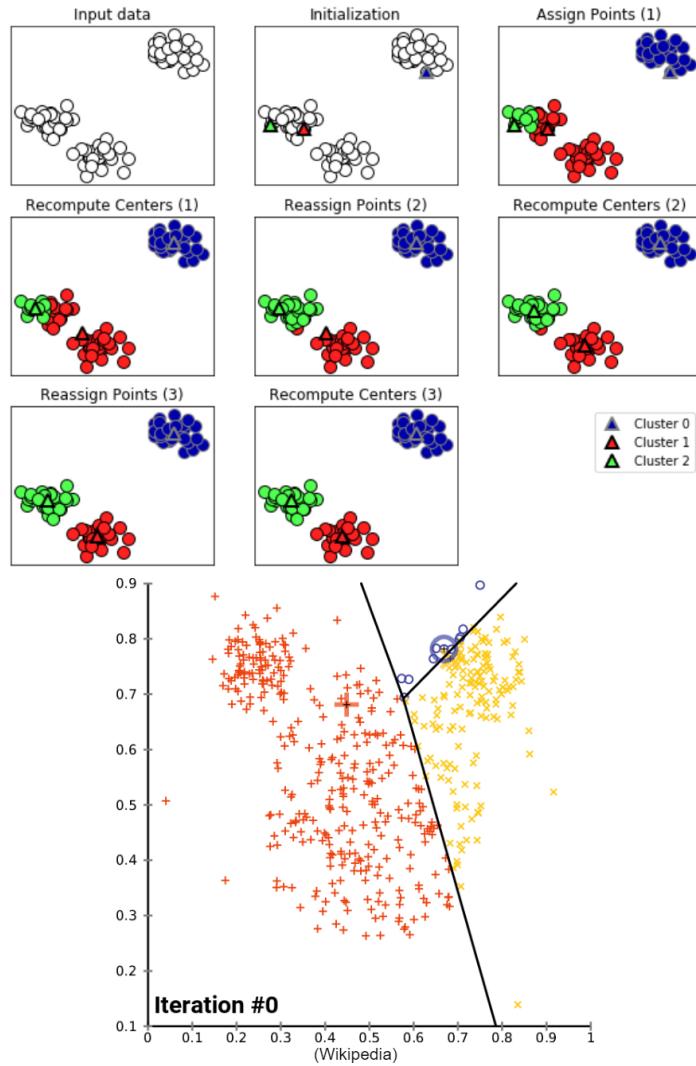
- K-means Clustering
- K-median Clustering
- K-mode Clustering

5.1 군집문제 해결을 위한 세팅 및 추정

1) 알고리즘 함수세팅: 군집문제를 푸는 대표적인 알고리즘 K-means Clustering

- 데이터의 영역을 대표하는 클러스터 중심 찾기
- 가장 단순하고 빠른 군집화 방법의 하나
- 비교적 이해하기 쉽고 구현도 쉬워 가장 인기 있는 군집 알고리즘

- (1) **Initialization:** 임의 공간 및 갯수의 클러스터 중심을 할당
- (2) **Assignment:** 클러스터 중심으로부터 모든 데이터에 대해 가까운 클러스터로 군집화 한다
- (3) **Update:** 각 클러스터 내부 데이터들의 평균으로 중심 재지정
- (4) **Finalization:** 클러스터 중심이 변화가 없거나 비용함수가 일정 수치 이하가 될 때까지 Assignment & Update를 반복



2) 함수 추정을 위한 비용함수:

"특정 출력(종속변수)/입력(독립변수)의 구분이나 관계 추론도 없고 학습을 위한 목표값도 없이, 주어진 데이터의 레이블을 할당하면서 유사한 그룹으로 군집화(Clustering) 하는 알고리즘"

- 이슈: 군집의 정답레이블이 없어 지도학습과 달리 명확한 모델링 평가 쉽지 않음
- 방향: 군집 내부 유사성을 높임 + 외부 군집들 간의 유사성을 낮춤

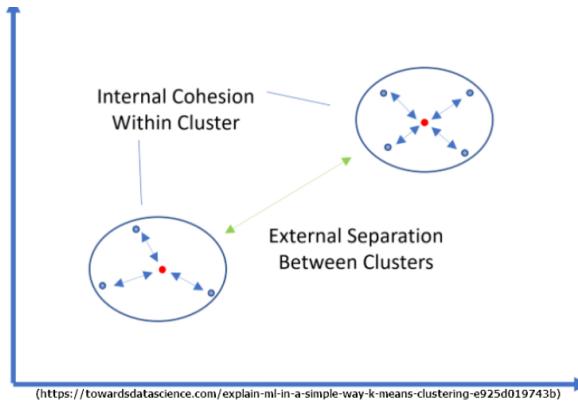
"응집도와 분리도를 모두 만족하는 군집을 찾는 최적화 과정"

• **응집도(Cohesion):** 군집 중심과 내부 데이터들의 유사성(거리) 합이 최소화

- 군집 내 분산(Inner-cluster Variance) 최소화

• **분리도(SepARATION):** 군집들 간의 비유사성 합이 최대화

- 군집 간 분산(Inter-cluster Variance) 최대화



- 유사성 측정:

- 컴퓨터의 데이터 분류:

대분류	소분류	컴퓨터의 분류1	컴퓨터의 분류2
질적변수(Qualitative Variable)	-	-	범주형
명목형 변수(Nominal Variable)	문자	범주형	
순위형 변수(Ordinal Variable)	숫자	범주형	
양적변수(Quantitative Variable)	-	-	연속형
이산형 변수(Discrete Variable)	숫자	연속형	
연속형 변수(Continuous Variable)	숫자	연속형	

- 측정 지표:

변수 종류	측정	설명
Continuous Variable	Manhattan Distance(Minkowski at Rank=1)	최단 루트 측정(변수들의 단위가 다르거나 상관성이 있으면 크게 변함)
	Euclidean Distance(Minkowski at Rank=2)	최단 거리 측정(변수들의 단위가 다르거나 상관성이 있으면 크게 변함)
	Standardized Distance	변수의 분산을 고려하여 표준화 측정
	Mahalanobis Distance	변수의 표준화 및 변수들의 상관관계 측정
	Weighted Euclidean Distance	Euclidean & Standardized의 일반화 측정
Continuous/Discrete Variable	Pearson's Correlation Coefficient	상관관계 측정
Discrete(Binary)/Nominal Variable	Simple Matching Coefficient	수식 참고
	Jaccard's Coefficient	수식 참고
	Russell and Rao Coefficient	수식 참고
Nominal Variable	Cosine Distance	문자 벡터들의 각도 측정
	Levenshtein Metric	문자 벡터들에서 다른 단어로 변경시 필요한 편집수 측정
	Tanimoto Coefficient(Expanded Jaccard's Coefficient)	문자 벡터 적용 Jaccard's Coefficient
Ordinal Variable	Rank Correlation Coefficient	순위기반 상관관계 측정
Continuous/Discrete/Nominal/Ordinal	Hamming Distance	같은 길이의 데이터에 같은 위치에 있는 값들의 비교 측정

- 비용함수: K-means Clustering 은 Euclidean Distance 를 비용함수로 최적화

$$Cost = \sum_{i=1}^N \min_{\mu_j \in C} (\|x_i - \mu_j\|^2)$$

where C is the estimated cluster,
and μ_j is the average center of j -th cluster.

3) K-centroid Clustering:

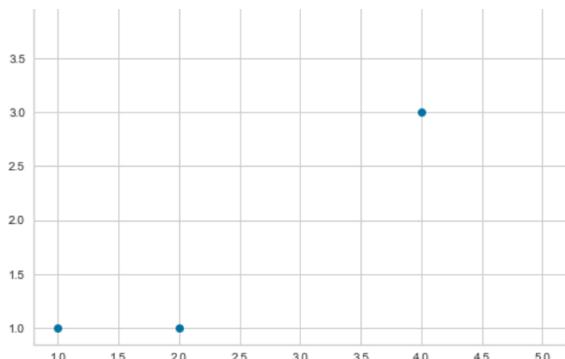
- K-means Clustering : 비용함수에서 평균 사용 군집화
- K-median Clustering : 비용함수에서 중앙값(Median) 사용 군집화
- 연속형 데이터: K-means Clustering 또는 K-median Clustering
- 이산형 데이터: K-mode Clustering 또는 K-medoid Clustering

5.2 군집문제 해결: K-means & Euclidean Distance

```
In [4]: import matplotlib.pyplot as plt
plt.plot([1,2,4,5], [1,1,3,4], 'o')
plt.show()
```

executed in 106ms, finished 23:57:12 2022-04-22





1) 반복1:

(1) **Initialization:** 데이터 중 임의 2개 중심 초기값 선택 - (1,1), (2,1)

(2-1) **Assignment:** Euclidean 거리 계산 - $\sqrt{(x_{\text{중심}} - x)^2 + (y_{\text{중심}} - y)^2}$

중심 초기값	샘플1	샘플2	샘플3	샘플4
(1,1)	0	1	3.6	5
(2,1)	1	0	2.8	4.2

(2-2) **Assignment:** 클러스터 확인 - {샘플₁}, {샘플₂, 샘플₃, 샘플₄}

중심 초기값	샘플1	샘플2	샘플3	샘플4
(1,1)	0.0	1.0	3.6	5.0
(2,1)	1.0	0.0	2.8	4.2
클러스터 중심	(1,1)	(2,1)	(2,1)	(2,1)

(3) **Update:** 중심 초기값 재지정 - (1,1), (3.7, 2.7)

$$\hat{\mu}_1(1) = \text{샘플}_1 = (1, 1)$$

$$\hat{\mu}_2(1) = \frac{\text{샘플}_2 + \text{샘플}_3 + \text{샘플}_4}{3} = (3.7, 2.7)$$

2) 반복2:

(2-1) **Assignment:** Euclidean 거리 계산 - $\sqrt{(x_{\text{중심}} - x)^2 + (y_{\text{중심}} - y)^2}$

중심 초기값	샘플1	샘플2	샘플3	샘플4
(1,1)	0.0	1.0	3.6	5.0
(3.7,2.7)	3.1	2.4	0.5	1.9

(2-2) **Assignment:** 클러스터 확인 - {샘플₁, 샘플₂}, {샘플₃, 샘플₄}

중심 초기값	샘플1	샘플2	샘플3	샘플4
(1,1)	0.0	1.0	3.6	5.0
(3.7,2.7)	3.1	2.4	0.5	1.9
클러스터 중심	(1,1)	(1,1)	(3.7,2.7)	(3.7,2.7)

(3) **Update:** 중심 초기값 재지정 - (1.5,1), (4.5,3.5)

$$\hat{\mu}_1(2) = \frac{\mathbf{x}_1 + \mathbf{x}_2}{2} = (1.5, 1)$$

$$\hat{\mu}_2(2) = \frac{\mathbf{x}_3 + \mathbf{x}_4}{2} = (4.5, 3.5)$$

3) 반복3:

(2-1) **Assignment:** Euclidean 거리 계산 - $\sqrt{(x_{\text{중심}} - x)^2 + (y_{\text{중심}} - y)^2}$

중심 초기값	샘플1	샘플2	샘플3	샘플4
(1.5,1)	0.5	0.5	3.2	4.6
(4.5,3.5)	4.3	3.5	0.7	0.7

(2-2) **Assignment:** 클러스터 확인 - {샘플₁, 샘플₂}, {샘플₃, 샘플₄}

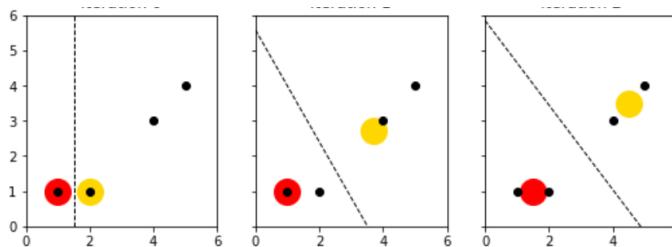
중심 초기값	샘플1	샘플2	샘플3	샘플4
(1.5,1)	0.5	0.5	3.2	4.6
(4.5,3.5)	4.3	3.5	0.7	0.7
클러스터 중심	(1.5,1)	(1.5,1)	(4.5,3.5)	(4.5,3.5)

(4) **Finalization:** 클러스터 중심의 변화가 없으므로 알고리즘 종료 - {샘플₁, 샘플₂}, {샘플₃, 샘플₄}

Iteration 0

Iteration 1

Iteration 2



5.3 군집문제 한계

1) 비지도학습 문제임에도 불구하고 K-means는 클러스터의 갯수 k를 미리 지정해야 함

- k 갯수에 따라서 결과는 천차만별이지만 k 갯수를 추정하지 못하는 것이 결정적 한계
- k 갯수 추론을 위한 별도의 방법론을 고려해야 함 (Elbow Method, Silhouette Method)

```
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer
model = KMeans()
visualizer = KElbowVisualizer(model, k=(1,10))
visualizer.fit(df)
```

2) 처음에 정하는 초기값에 따라서 결과도 달라질 수 있고 알고리즘 성능에 영향을 줌

- 고객 세분화(Segmentation)에서 매번 초기값에 따라 묶이는 고객 그룹이 달라질 수 있음
- 잘못된 초기값은 무한/많은 반복과정을 유발시키거나 군집 성능을 떨어뜨릴 수 있음
- 적절한 초기값을 결정하기 위한 방법론을 고려해야 함 (K-means++)

- (1) K-means++는 임의 공간 중심 초기값 대신 데이터들 중 무작위로 1개의 중심 초기값 선택
- (2) 기 지정된 초기값에서 최대한 먼 곳에 배치된 데이터를 다음 중심 초기값으로 지정
- (3) k 개의 중심 초기값이 선택될 때까지 위 과정을 반복

- K-means++ 방식 적용시, 기준보다 알고리즘의 반복이 줄어들어 속도가 빨라지고 성능이 좋아짐

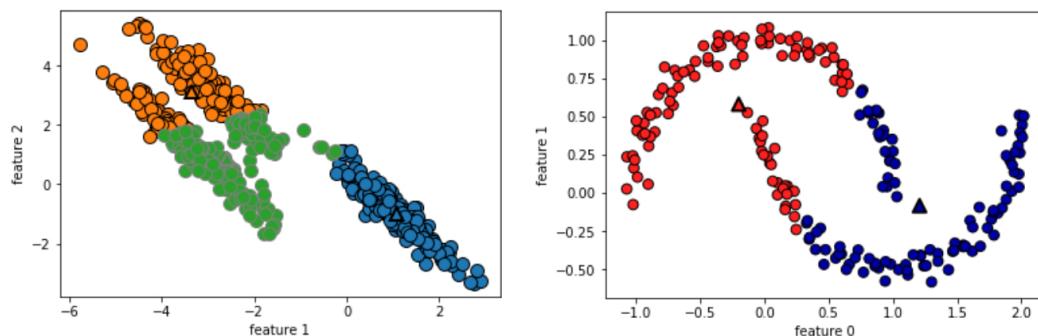
```
from sklearn.cluster import KMeans
model = KMeans(n_clusters=k, init='k-means++')
```

3) 구분된 군집이 일부 데이터에선 최적일 수 있으나 전체 데이터에선 최적(Global Optima)을 보장하지 않는 지역 최적화 알고리즘(Local Optimization Algorithm)

4) 평균계산 방식이라 일부 Outlier로 인해 결과가 달라질 수 있음

- 평균 보다 중앙값이 이상치에 덜 민감하므로 K-median Clustering이 K-means Clustering의 대안이 될 수 있음
- 데이터 탐색 과정에서 이상치 제거하는 것도 방법

5) Euclidean Distance 방식이라 원형/구형 클러스터 기준으로 군집을 구분하기 때문에, 다른 도넛형, 사각형, 핵비공형 등 의 다양한 모양이나 다양한 밀도의 클러스터 인식에는 효과가 없음



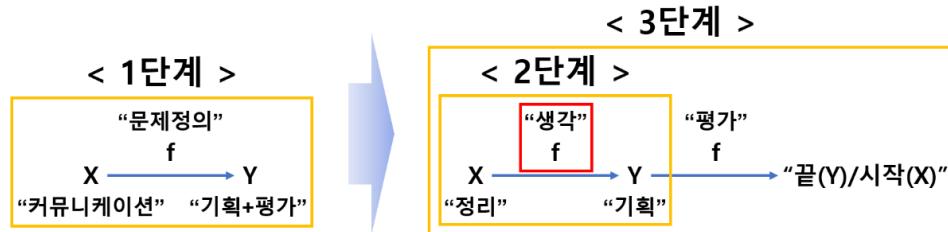
6) Hard Clustering 방식이라 여러 클러스터에 포함될 수 있는 데이터를 고려하지 못함

5.4 Mini-batch K-means Clustering

- K-means Clustering은 중심위치와 모든 데이터 사이의 거리를 계산해야 하기 때문에 데이터의 갯수가 많아지면 계산량도 늘어남
- 데이터의 수가 너무 많을 때는 미니배치 K-평균(Mini-batch) 군집화 방법을 사용하면 계산량을 줄일 수 있음
- 데이터를 미니배치 크기만큼 무작위로 분리하여 K-means Clustering
- 모든 데이터를 사용했을 경우와 결과가 다를 수 있지만 큰 차이 없음
- scikit-learn의 cluster 패키지는 MiniBatchKMeans 클래스를 제공하여 큰 대용량 데이터셋에서도 잘 작동

```
from sklearn.cluster import KMeans, MiniBatchKMeans
model = KMeans(n_clusters=k, init='k-means++')
# 대신
model = MiniBatchKMeans(n_clusters=k, batch_size=1000)
```

6 검증지표 방향(Evaluation Metrics)



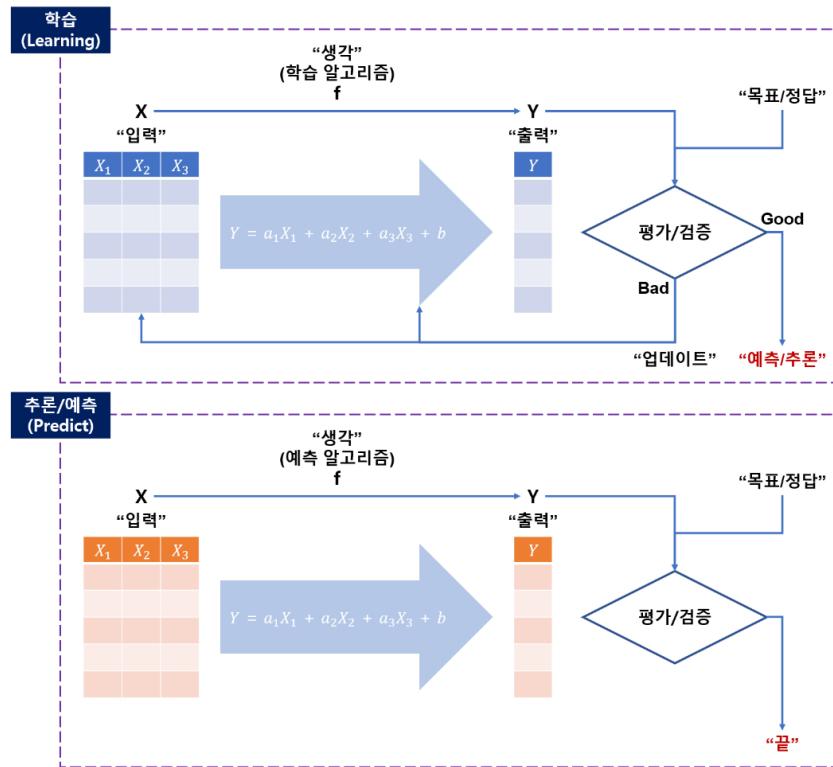
“문제해결 검증지표 와 알고리즘 검증지표 는 같을 수 있으나 대부분은 다른 편”

(1) 문제해결 검증지표: 실제 문제를 잘 해결하는지 평가 (3단계)

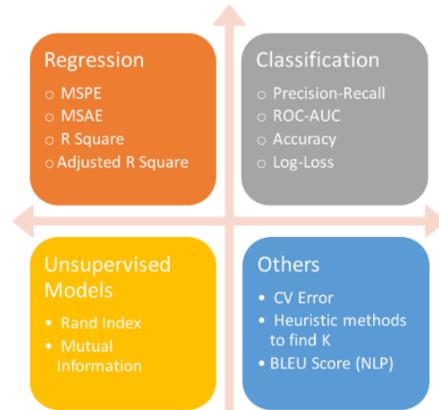
(2) 알고리즘 검증지표: 데이터의 패턴이 잘 추출되고 예측의 정확성을 평가 (2단계)

- 알고리즘 성능이 좋은 것과 문제해결이 가능한 것은 다르기 때문에 문제해결 지표와 알고리즘 지표는 대부분은 다른 편
- 알고리즘 검증지표는 없어도 되지만 문제해결 검증지표는 반드시 필요
- (이론적) 알고리즘 들은 일반적으로 특정 알고리즘 검증지표를 향상시키는 방향으로 개발됨

6.1 대표적인 검증지표



1) 문제별 종류:



- **Statistical Metrics:** Correlation
 - 입력(Input): -무한대 ~ 무한대 범위의 연속형 값
 - 출력(Output): 이론적으로 $-1 \sim 1$ 범위의 연속형 값
 - **Regression Metrics:** MSE, MSPE, RMSE, RMSLE, MAE, MAPE, MPE, R², Adjusted R², ... (Y의 범위가 무한대가 가능한 연속형일 때)
 - 입력(Input): -무한대 ~ 무한대 범위의 연속형 값
 - 출력(Output): 이론적으로 0 ~ 무한대 범위의 연속형 값
 - **Classification Metrics:** Log Loss, Cross-entropy, ROC, AUC, Gini, Confusion Matrix, Accuracy, Precision, Recall, F1-score, Classification Report, KS Statistic, Concordant-Discordant Ratio, (ARI, NMI, AMI), ... (Y가 2개 또는 그 이상개수의 이산형일때)
 - 입력(Input): -무한대 ~ 무한대 범위의 연속형 값
 - 출력(Output): 알고리즘 종류에 따라 출력이 달라질 수 있음
 - 확률(Probability): 0 ~ 1 범위의 연속형 값 (Logistic Regression, Random Forest, Gradient Boosting, Adaboost, ...)
 - 집단(Class): 0 또는 1 의 이산형 값 (SVM, KNN, ...)
 - **Clustering:** Dunn Index, Silhouette, ...
 - **Ranking Metrics:** Gain, Lift, MRR, DCG, NDCG, ...
 - **Computer Vision Metrics:** PSNR, SSIM, IoU, ...
 - **NLP Metrics:** Perplexity, BLEU score, ...
 - **Deep Learning Related Metrics:** Inception score, Frechet Inception distance, ...
 - **Real Problem:** ???

2) 검증지표 성능의 종류: 데이터/분석은 높은 정확도를 놓거나 높은 에러를 발생시킴

- **높은정확도(High Accuracy)**: 과거 패턴이 미래에도 그대로 유지가 된다면 예측 정확도가 높아짐
 - **높은에러(High Error)**: 패턴이 점차적으로 또는 갑자기 변형되면 예측값은 실제값에서 크게 벗어날 수 있음
 - **Black Swan**: 일어날 것 같지 않은 일이 일어나는 현상
 - **White Swan**: 과거 경험들로 충분히 예상되는 위기지만 대응책이 없고 반복될 현상
 - **Gray Swan**: 과거 경험들로 충분히 예상되지만 발생되면 충격이 지속되는 현상

6.2 군집분석 검증지표 및 해석하기

0) 방향:

- 군집 알고리즘을 적용하고 평가하는 것이 매우 정성적인 분석 과정
 - 레이블 정답이 없기 때문에 단순정확도(Accuracy) 등 지표로 사용 불가
 - 군집 레이블 뿐만 아니라 최적의 전체 군집의 수도 알아내기 쉽지 않음
 - 군집 탄당성 지표(Clustering Validity Index): 군집 결과의 유용성 평가

- (1) 군집간 거리
 - (2) 군집의 지름
 - (3) 군집의 분산

• 종류•

- Dunn Index
 - Silhouette Index
 - ARI(Adjusted Rand Index)
 - NMI(Normalized Mutual Information)
 - AMI(Adjusted Mutual Information)

1) Dunn 지표(Dunn Index): 군집 내 데이터 간 거리 최대값 대비 군집간 거리 최소값의 비율

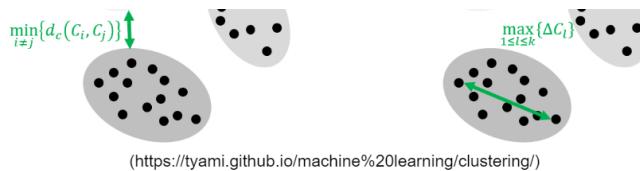
- 군집 간 거리는 멀수록 & 군집 내 분산은 작을수록 좋은 군집화이며 Dunn Index는 증가

$$DI(C) = \frac{\min_{i \neq j} \{d_c(C_i, C_j)\}}{\max_{1 \leq l \leq k} \{\Delta C_l\}}$$

Minimize Smallest distance between Clusters

Maximize Largest distance within a cluster





2) 실루엣 지표(Silhouette Index): 군집 내 데이터 간 거리 평균과 군집 외부 가장 가까운 군집 내 데이터와의 거리 평균 차이

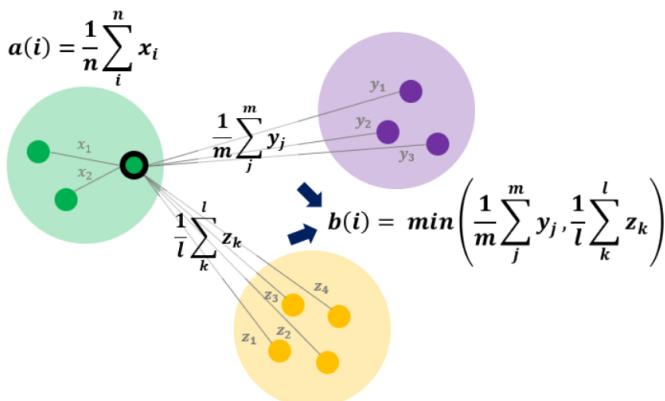
$$S(i) = \frac{b(i) - a(i)}{\max \{a(i), b(i)\}}$$

군집 내 응집도(Cohesion)

- $a(i)$: i 번째 군집 내 속한 데이터들과의 거리 평균

군집 간 분리도(Separation)

- $b(i)$: i 번째 군집과 가장 가까운 군집에 속한 데이터들과의 거리 평균

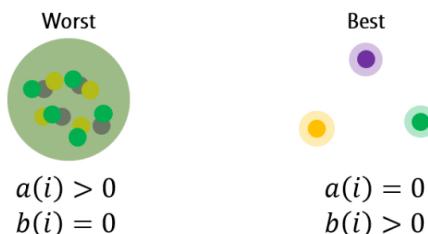


(https://tyami.github.io/machine%20learning/clustering/)

- 보통 실루엣 지표가 0.5보다 크면 군집 결과가 타당한 것으로 평가

$$-1 < s(i) < 1$$

Heuristic: $s(i) = 0.5$



$$\therefore s(i) = -1$$

$$\therefore s(i) = 1$$

(https://tyami.github.io/machine%20learning/clustering/)

- 오히려 평가보다 군집 갯수를 결정하는데 많이 사용
- 밀집된 클러스터에선 성능이 좋으나 모양이 복잡할 때는 평가 성능 좋지 않음

3) Others: 클러스터 레이블 정답을 아는 경우 군집성능 평가

"일부 라도 레이블이 있는 경우, 알고리즘 성능 체크에 주로 사용"

(1) ARI(Adjusted Rand Index): 얼마나 많은 클러스터들이 정답과 유사한지 측정

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

(2) NMI(Normalized Mutual Information): 상관관계 계계를 대체하기 위해 실제와 예측 클러스터 생성을 위한 정보량 분포의 유사성/의존도 측정

```
from sklearn.metrics import normalized_mutual_info_score
```

(3) AMI(Adjusted Mutual Information): 클러스터 수가 많을 때 NMI가 높아지는 한계를 보완

```
from sklearn.metrics import adjusted_mutual_info_score
```

4) 현실 지표:

- 데이터에 잡음/변화를 주었을 때의 결과 변동성 고려

- 알고리즘 매개변수에 변화를 주었을 때의 결과 변동성 고려
- 여러가지 변화에도 결과가 일정하면 (Robust) 신뢰할만한 모델링