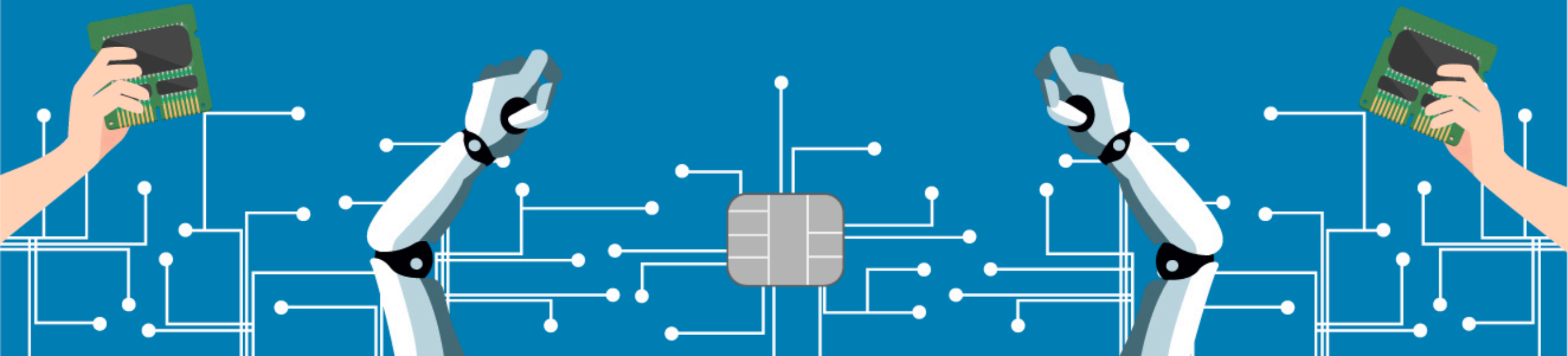
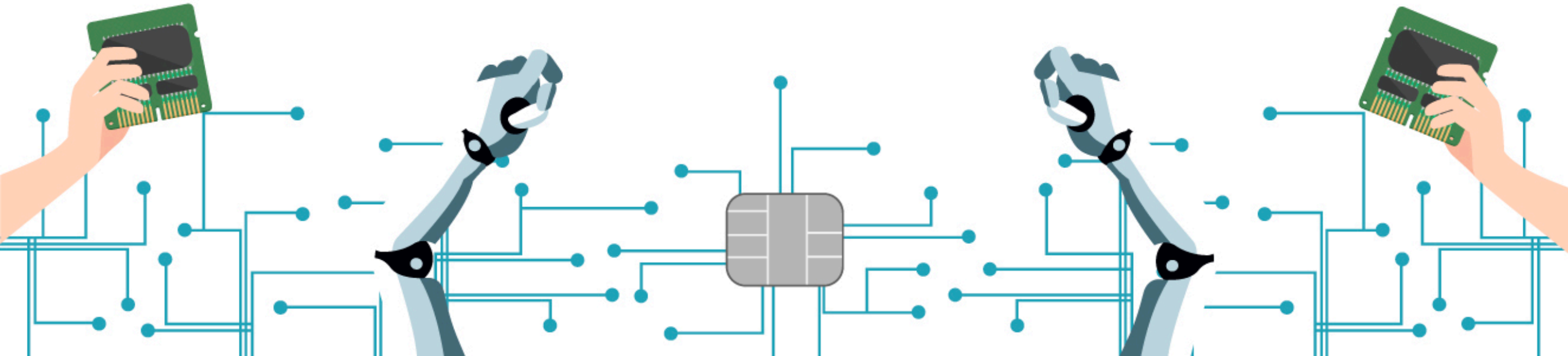
Kuggle

2022_2_22 Kuggle 방학정규세션_W8

2022.2.22



CONTENTS



CONTENTS

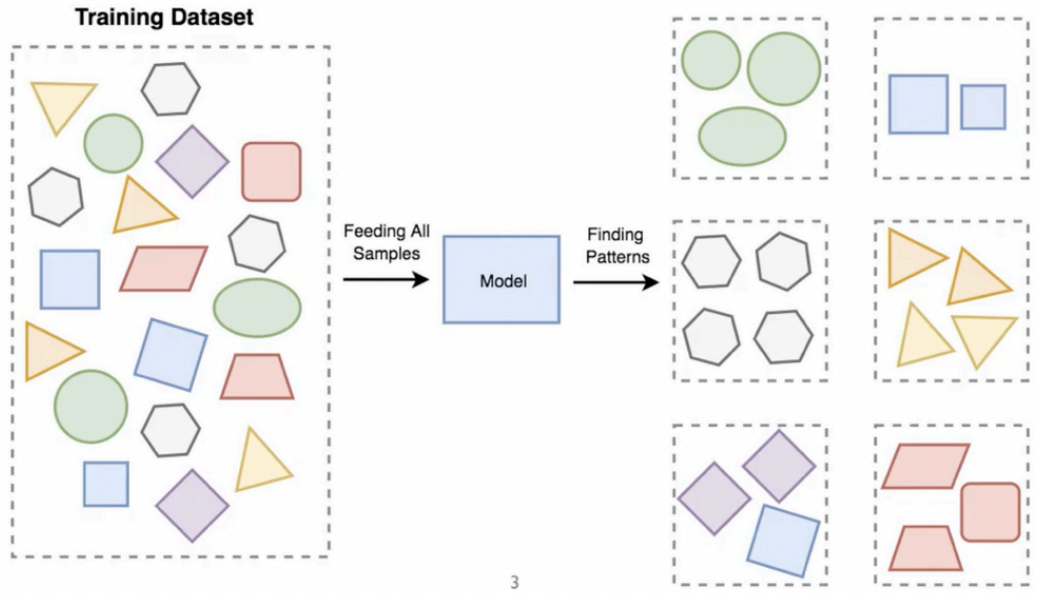
01. Overview

☑ Supervised 🆚 Unsupervised

01 Overview

-Supervised VS Unsupervised

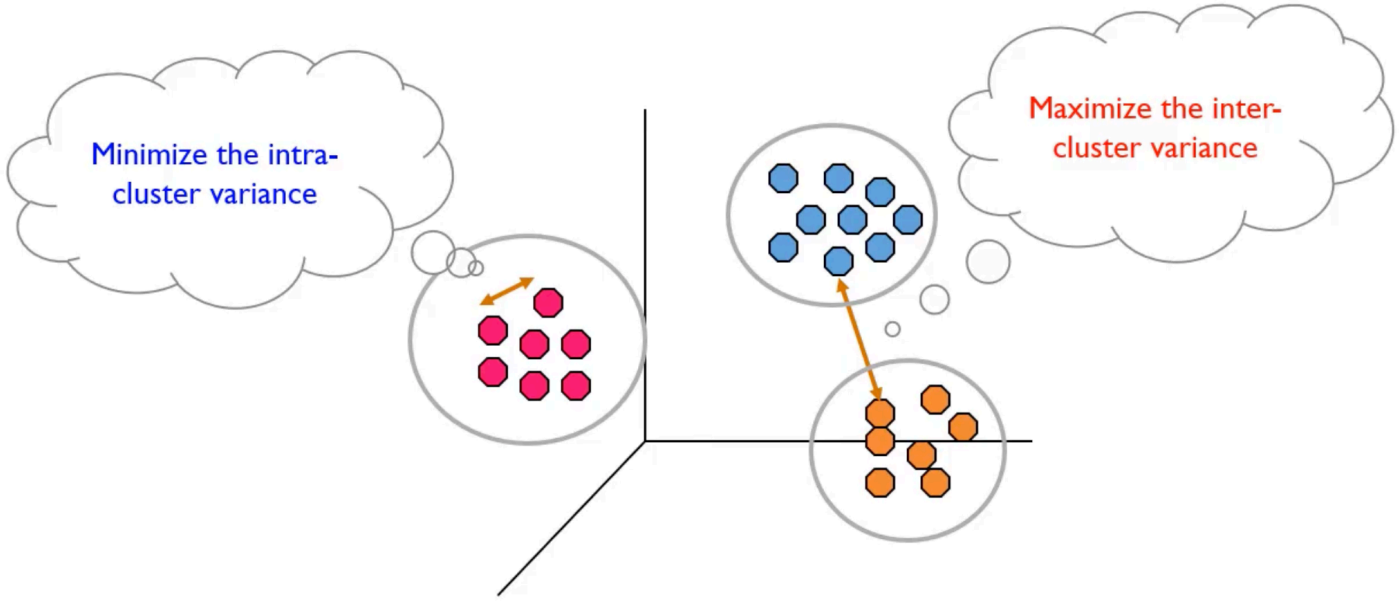
- Supervised vs. Unsupervised Learning
  - ✓ Supervised: Find a function f that explains the relationship between the input X and the output Y
  - ✓ Unsupervised: Explore the features of the input X



01 Overview

-Supervised VS Unsupervised

- What is clustering?
  - ✓ Find groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



CONTENTS

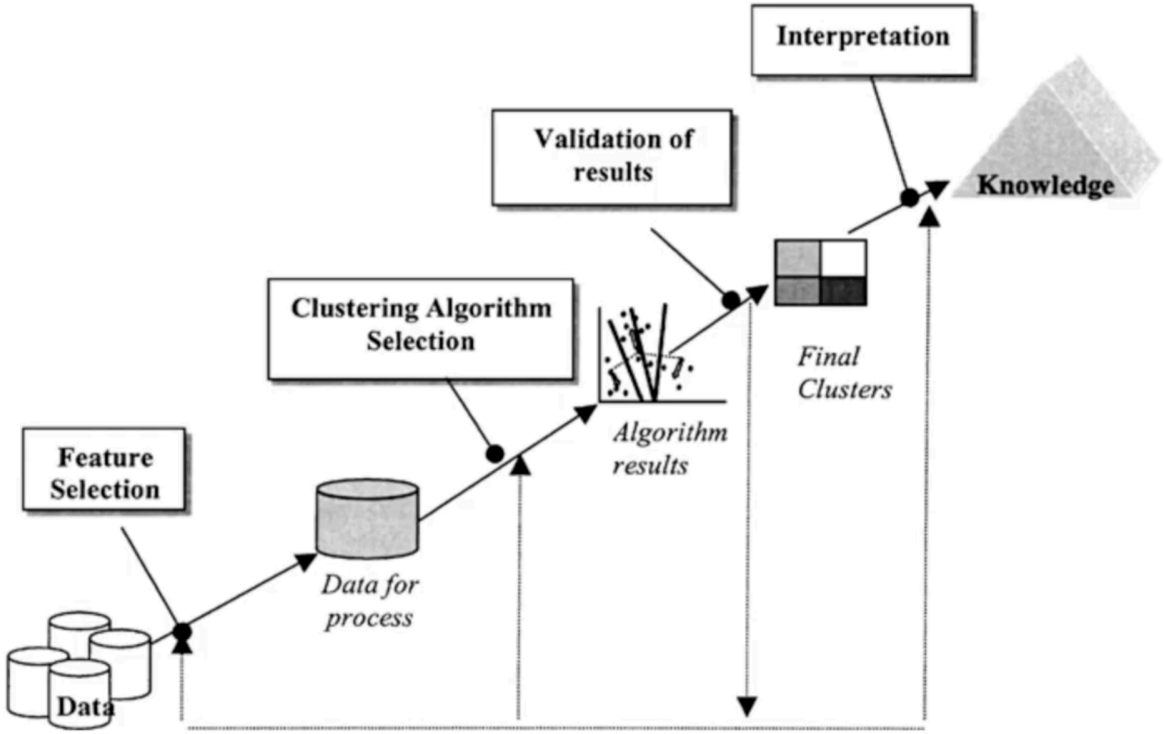
02. 군집 평가

- ☑ Overview
- ☑ Silhouette Coefficient
- ☑ Elbow Method

02 군집 평가

-Overview

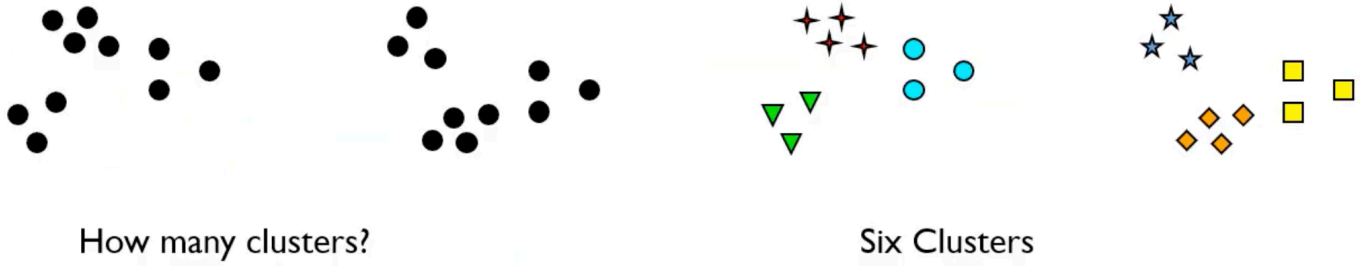
- Standard clustering procedure



02 군집 평가

-Overview

- How many clusters are optimal?



02 군집 평가

-Overview

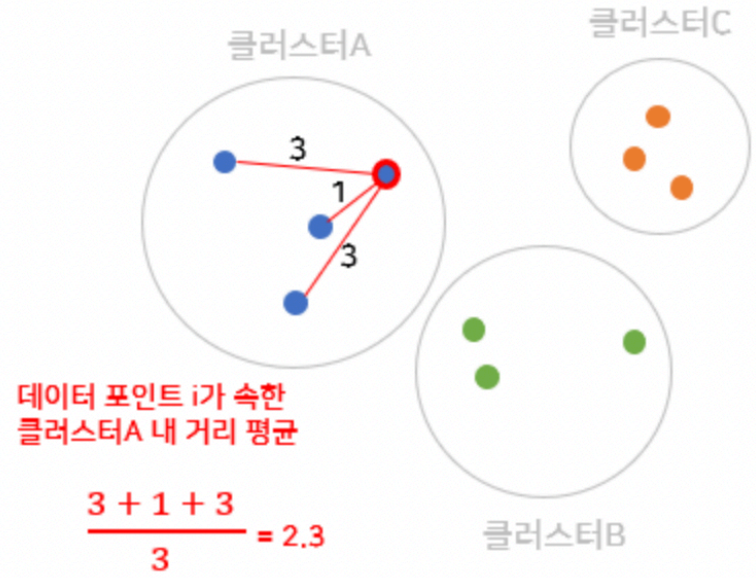
- Examples of clustering validity measures

External	Internal	Relative
Rand Statistic	Cophenetic Correlation Coefficient	Dunn family of indices
Jaccard Coefficient	Sum of Squared error (SSE)	Davies-Bouldin (DB) index
Folks and Mallows index	Cohesion and separation	Semi-partial R-squared
(Normalized) Hurbert Γ statistic		SD validity index
		Silhouette

02 군집 평가

-Silhouette Coefficient

- a(i)



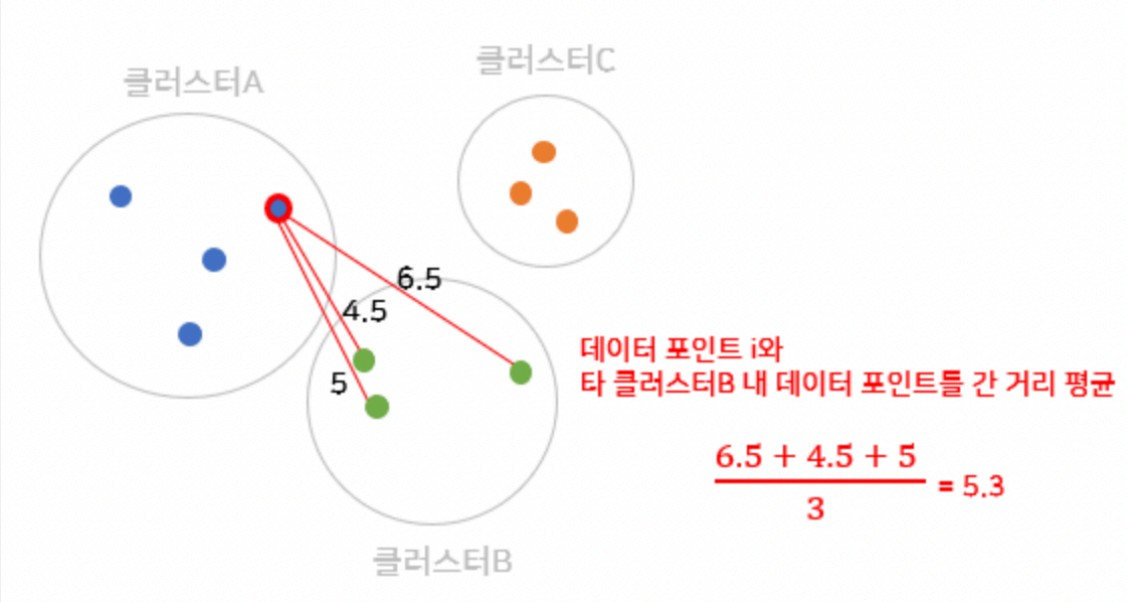
$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \quad -1 \le s(i) \le 1$$

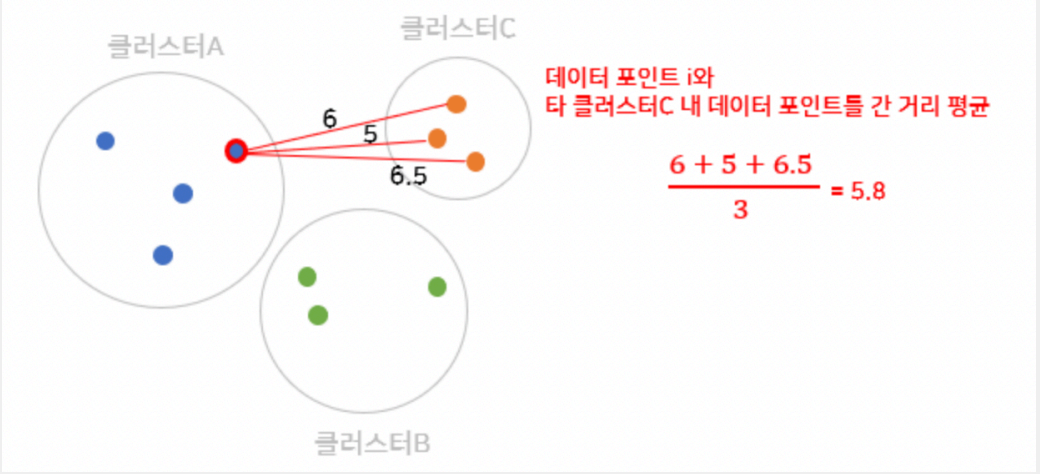
$a(i)$: 데이터 포인트 i가 속한 클러스터 내 데이터 포인트들과 거리 평균

02 군집 평가

-Silhouette Coefficient

- d(I, C)



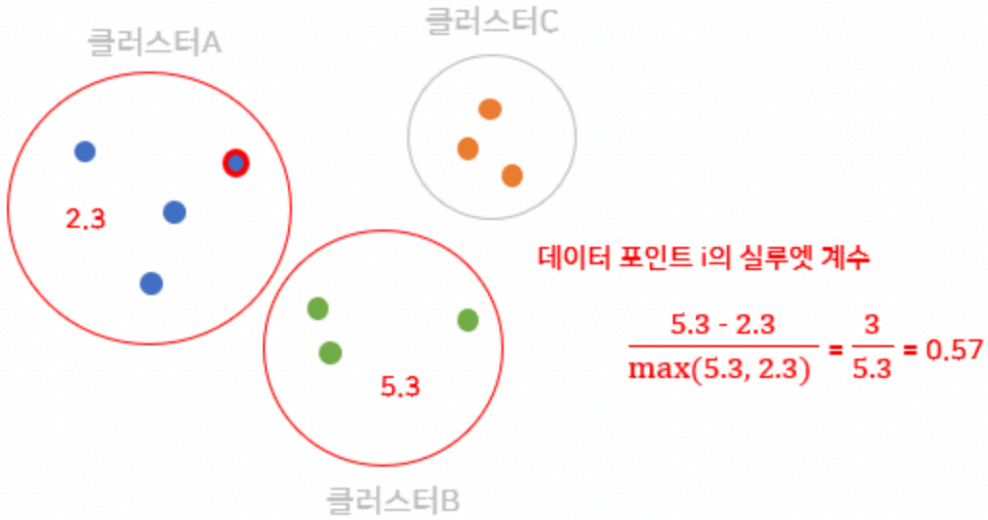


$d(i, C)$: 데이터 포인트 i가 속하지 않은 클러스터C 의 데이터 포인트들과 거리 평균

02 군집 평가

-Silhouette Coefficient

- s(i)

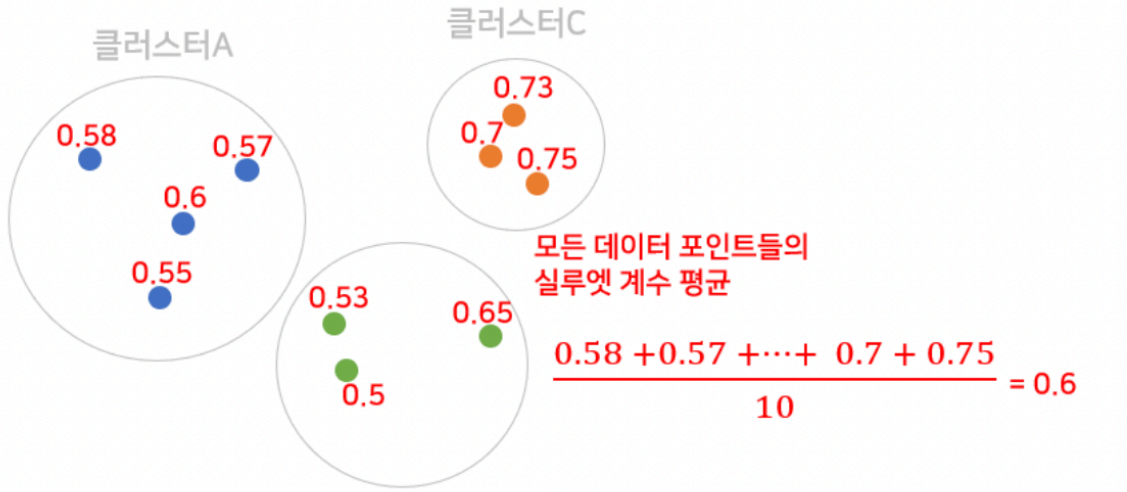


데이터 포인트 i의 실루엣 계수

$$\frac{5.3 - 2.3}{\max(5.3, 2.3)} = \frac{3}{5.3} = 0.57$$

$b(i)$: $d(i, C)$의 최솟값

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



모든 데이터 포인트들의
실루엣 계수 평균

$$\frac{0.58 + 0.57 + \cdots + 0.7 + 0.75}{10} = 0.6$$

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

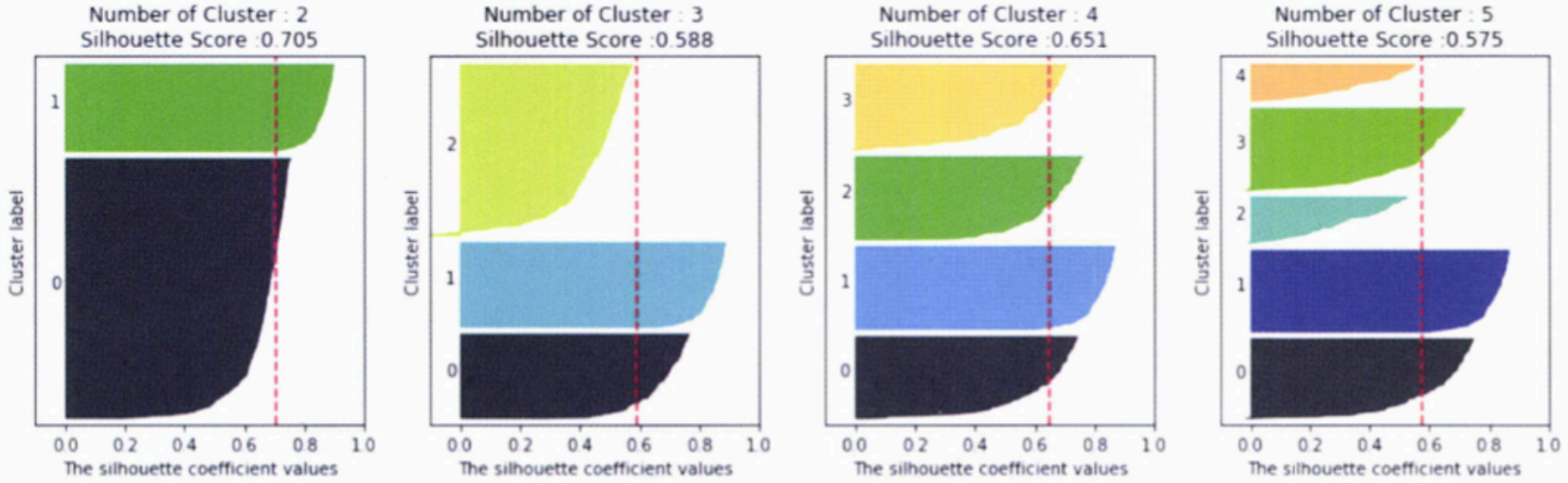
$$s(i) = \begin{cases} 1 - a(i)/b(i), & if\ a(i) < b(i) \\ 0, & if\ a(i) = b(i) \\ b(i)/a(i) - 1, & if\ a(i) > b(i) \end{cases}$$

02 군집 평가

-Silhouette Coefficient

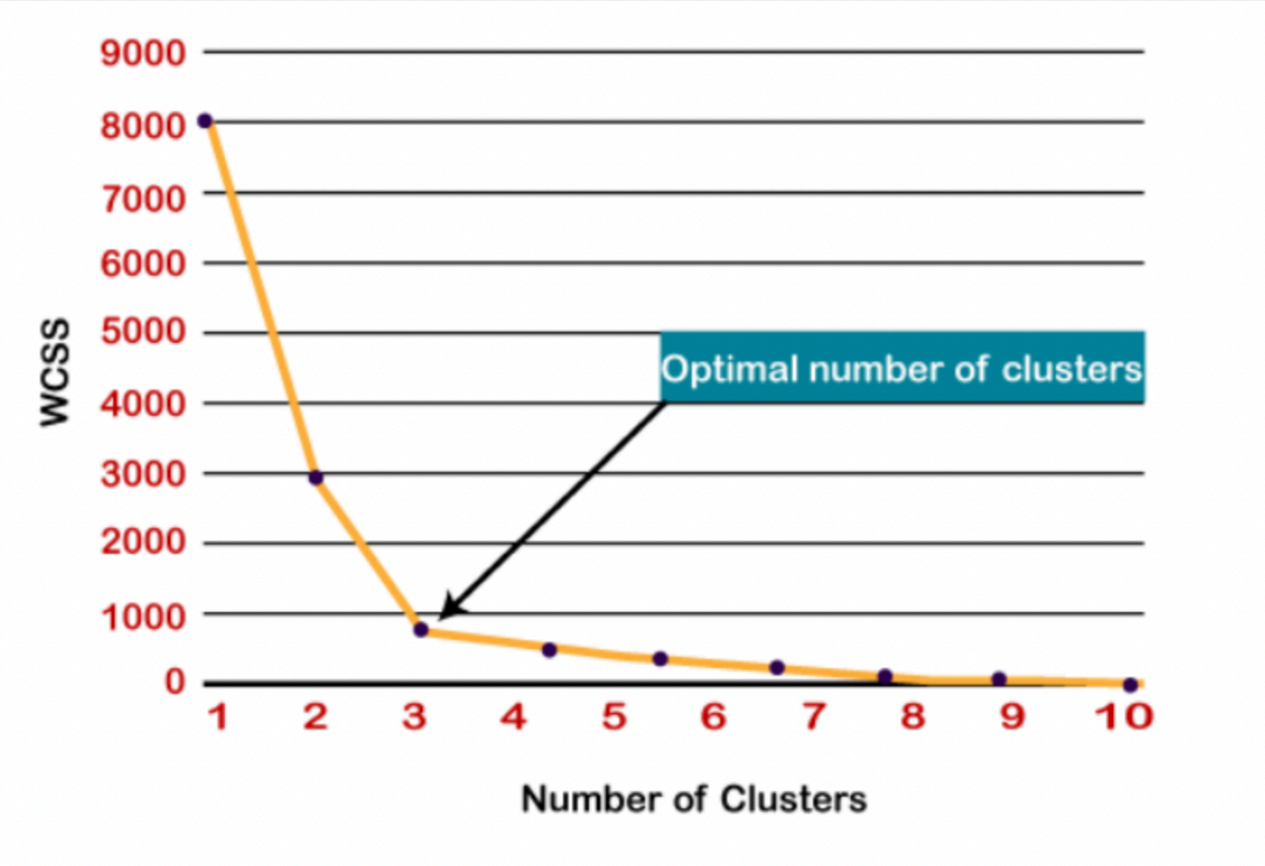
```
class sklearn.cluster.KMeans(n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001,
                             precompute_distances='auto', verbose=0, random_state=None,
                             copy_x=True, n_jobs=1, algorithm='auto')
```

```
visualize_silhouette([ 2, 3, 4, 5], X)
```



02 군집 평가

-Elbow Method



WCSS = Within Clusters Sum of Squares
= 클러스터 내 총 변동

https://nicola-ml.tistory.com/66

CONTENTS

03. K-Means Clustering

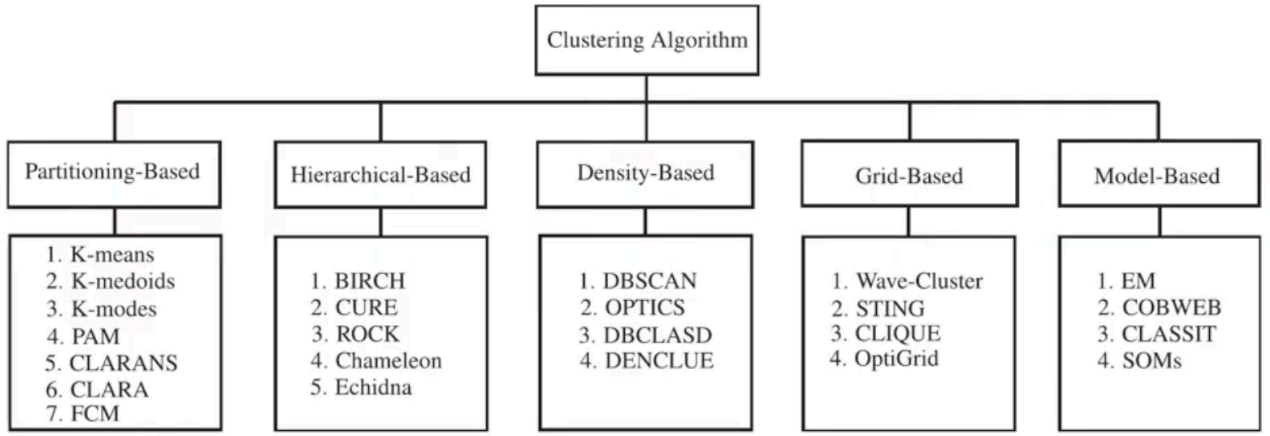
- ☑ Hard clustering 🆚 Soft clustering
- ☑ Partitional clustering 🆚 Hierarchical clustering
- ☑ K-Means Clustering Process
- ☑ Effect of initial centroids
- ☑ Limitations of K-Means Clustering

03 K-Means Clustering

-Hard clustering 🆚 Soft clustering

✓Hard Clustering (Crisp Clustering)

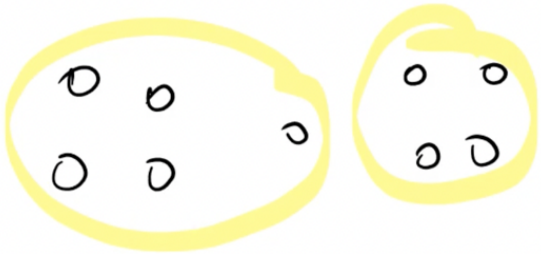
- Results in non-overlapping clusters
- Each instance belongs to only one cluster



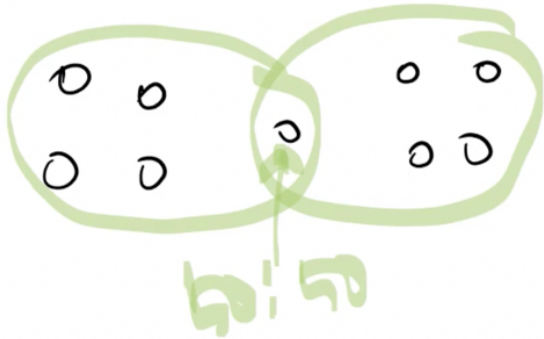
✓Soft Clustering (Fuzzy Clustering)

- Possible to result in overlapping clusters
- Each instance can belong to more than two clusters

👉 Hard Clustering



👉 Soft Clustering

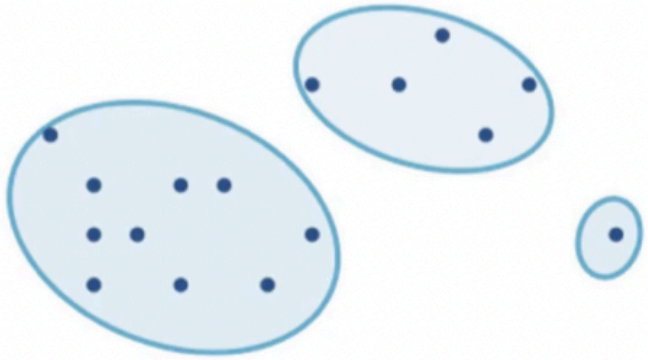


03 K-Means Clustering

-Partitional clustering VS Hierarchical clustering

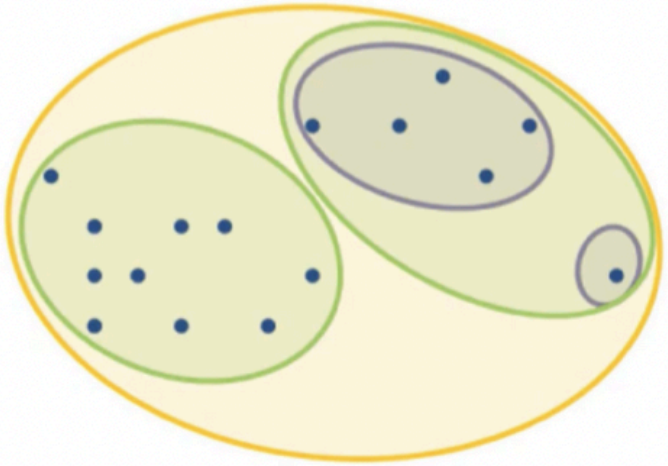
- Partitional clustering
  - ✓ Divide data into non-overlapping subsets such that each data object is in exactly one subset

Partitional Clustering



- Hierarchical clustering
  - ✓ A set of nested clusters organized as a hierarchical tree

Hierarchical Clustering



03 K-Means Clustering

-K-Means Clustering Process

- K-Means Clustering (KMC)
  - ✓ Partitional clustering approach
    - Each cluster is associated with a centroid
    - Each point is assigned to the cluster with the closest centroid
    - Number of cluster, K, must be specified

$$\mathbf{X} = C_1 \cup C_2 \ldots \cup C_K, \quad C_i \cap C_j = \phi, \quad i \neq j$$

$$\arg\min_{\mathbf{C}} \sum_{i=1}^{K} \sum_{\mathbf{x}_j \in C_i} ||\mathbf{x}_j - \mathbf{c}_i||^2$$

03 K-Means Clustering

-K-Means Clustering Process

K-means algorithm

Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

Repeat {

for i = 1 to m

$c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$

for k = 1 to K

μ_k := average (mean) of points assigned to cluster k

}

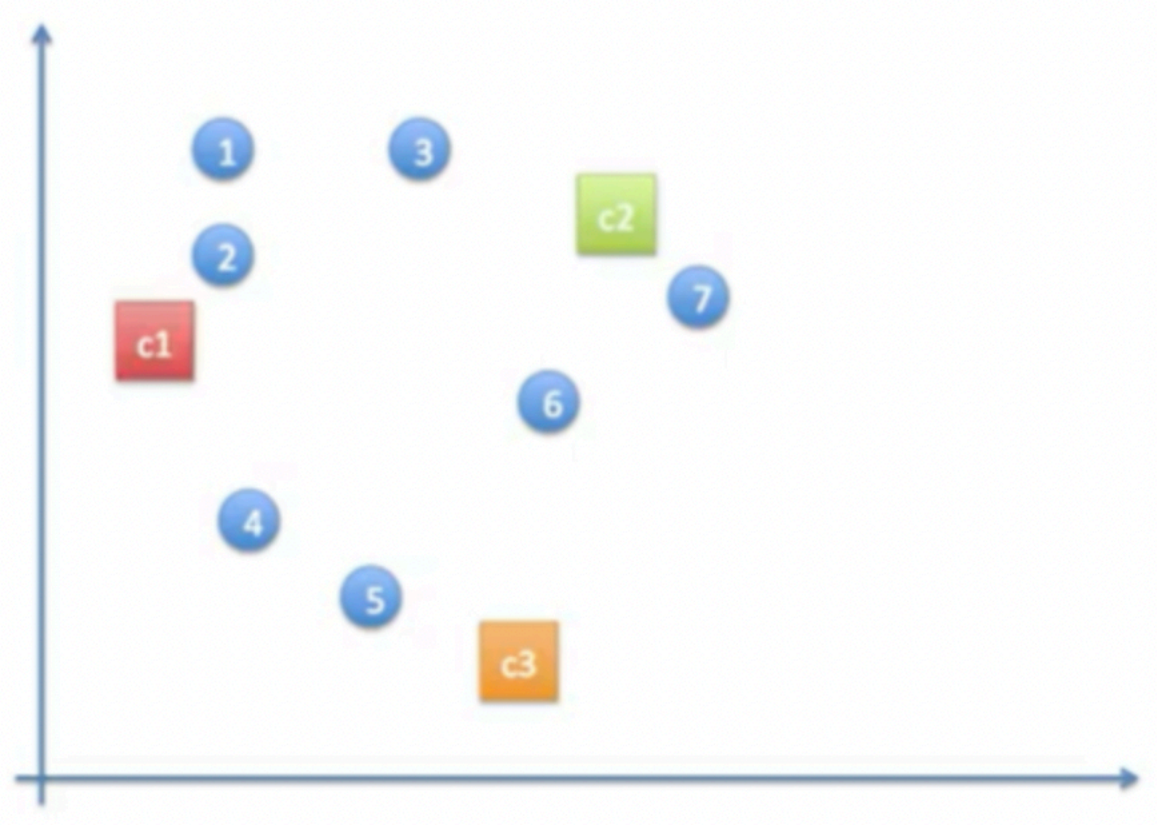
1 클러스터 할당 단계

2 클러스터 중심 이동 단계

03 K-Means Clustering

-K-Means Clustering Process

1. 클러스터의 중심(centroid)를 random하게 초기 설정



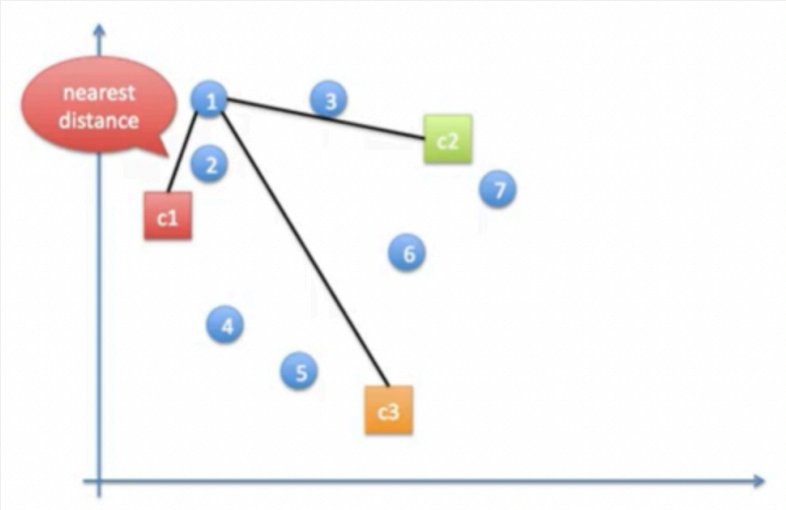
https://computer-science-student.tistory.com/entry/k-mean-클러스터링

03 K-Means Clustering

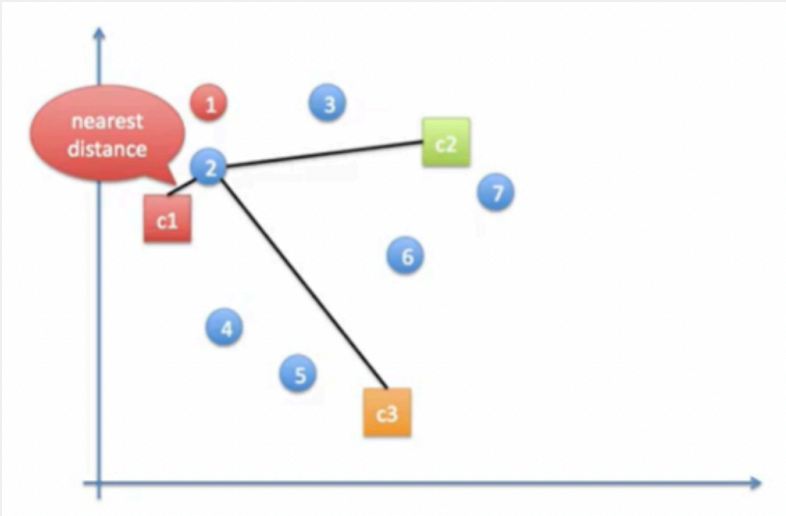
-K-Means Clustering Process

2. 각 데이터 포인터들을 클러스터에 할당

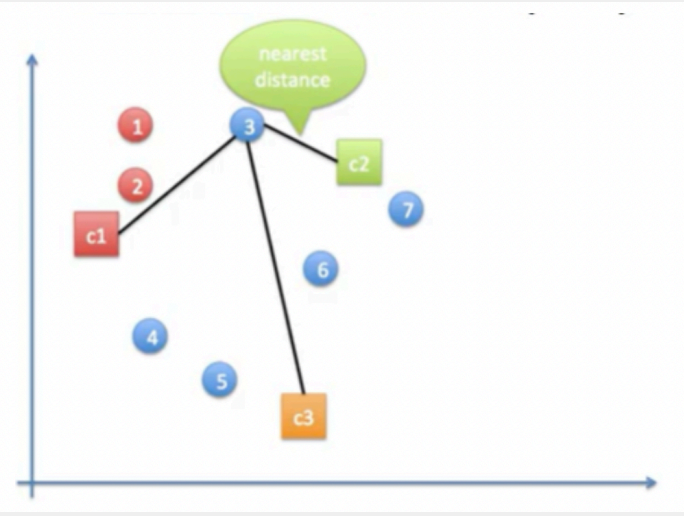
1번 데이터 포인트 할당



2번 데이터 포인트 할당



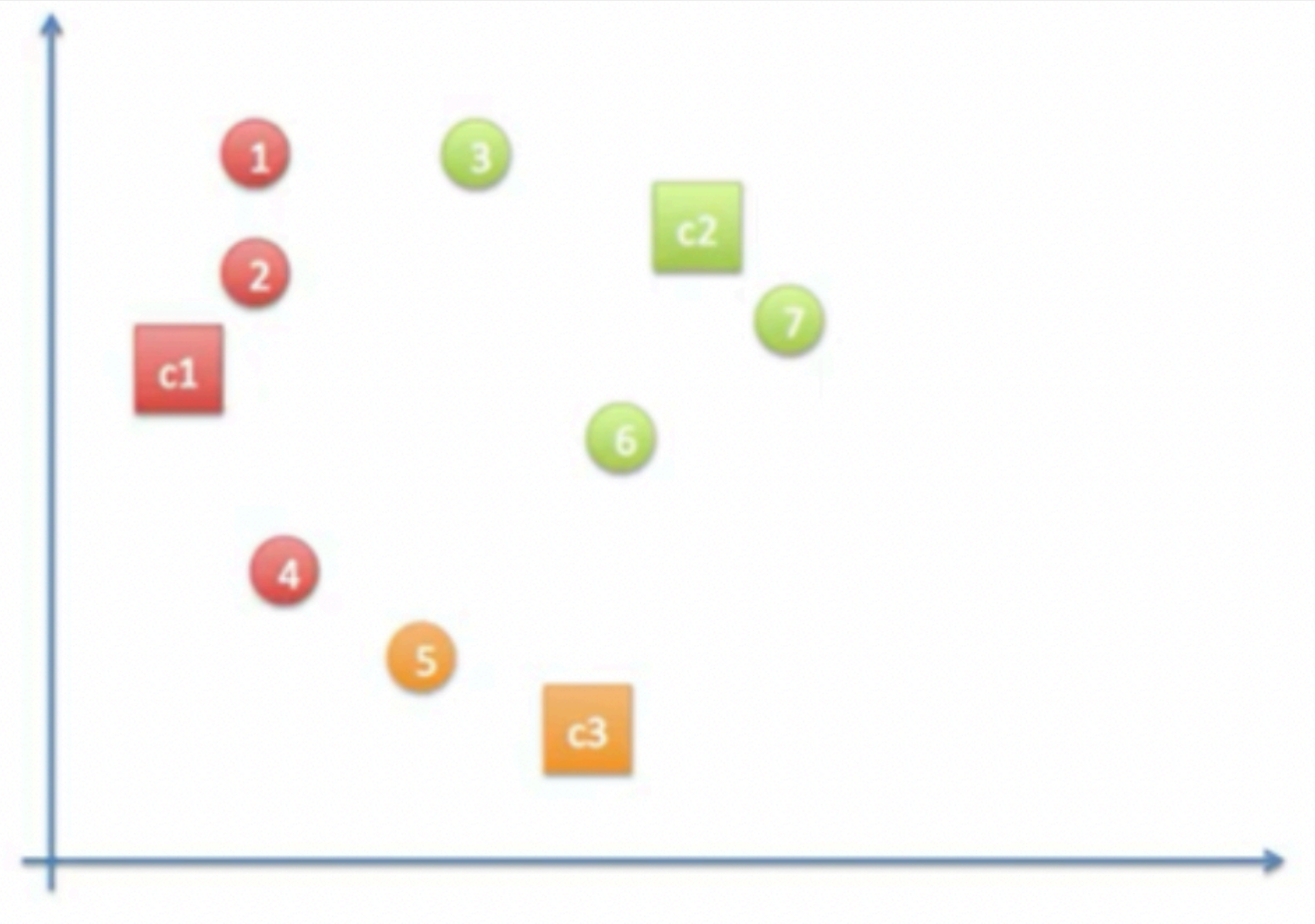
3번 데이터 포인트 할당



03 K-Means Clustering

-K-Means Clustering Process

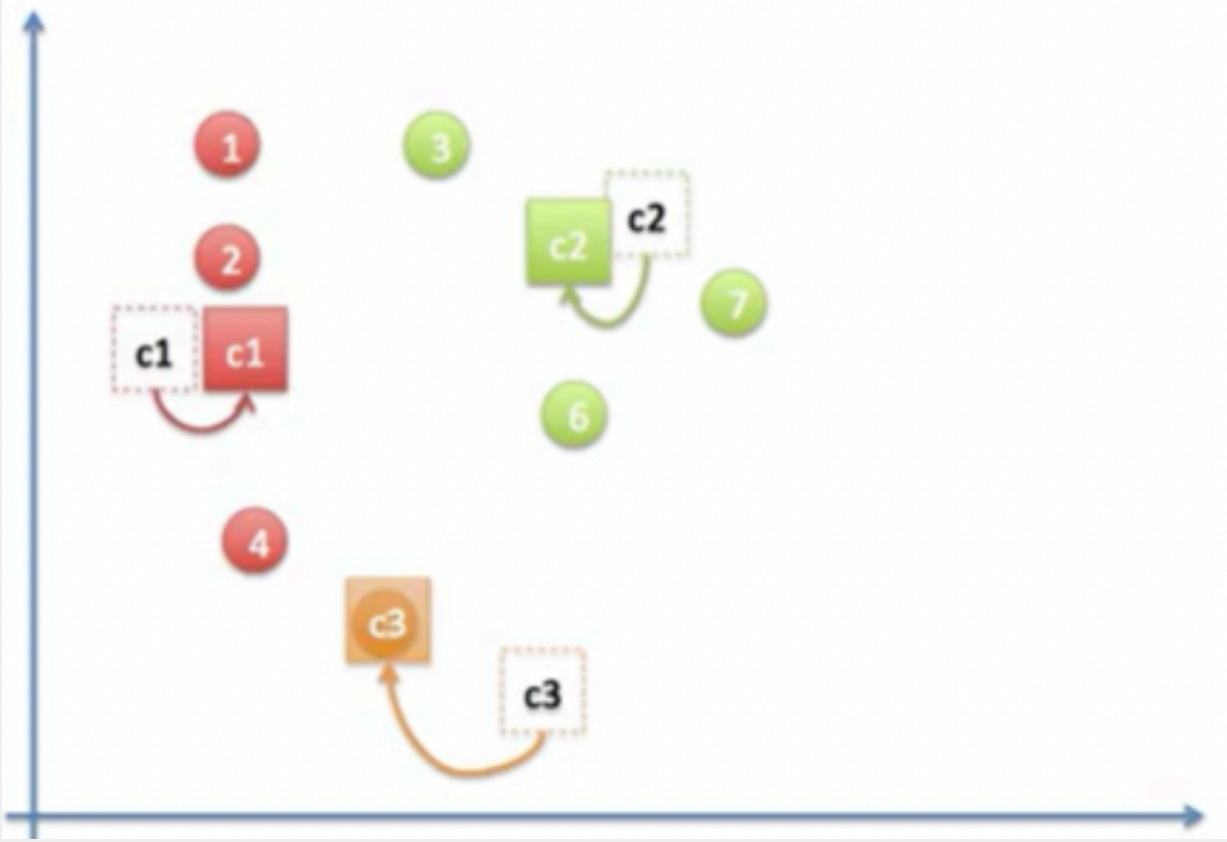
2. 클러스터 할당 결과



03 K-Means Clustering

-K-Means Clustering Process

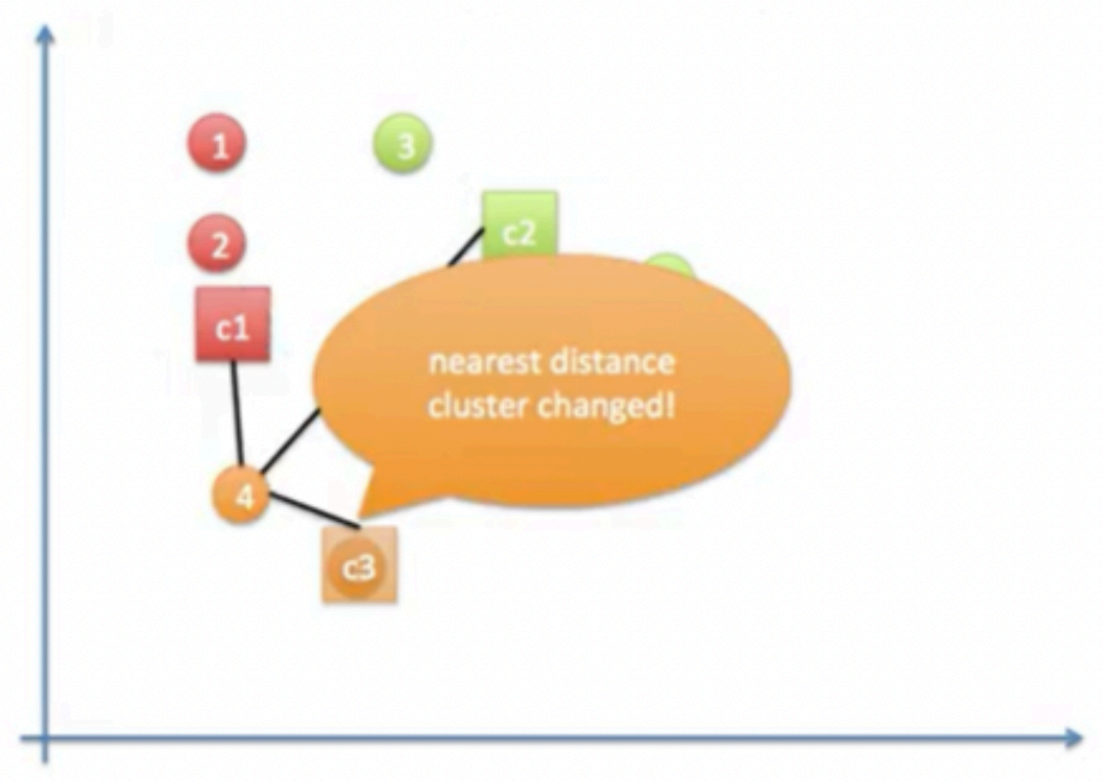
3. 클러스터 중심 이동



03 K-Means Clustering

-K-Means Clustering Process

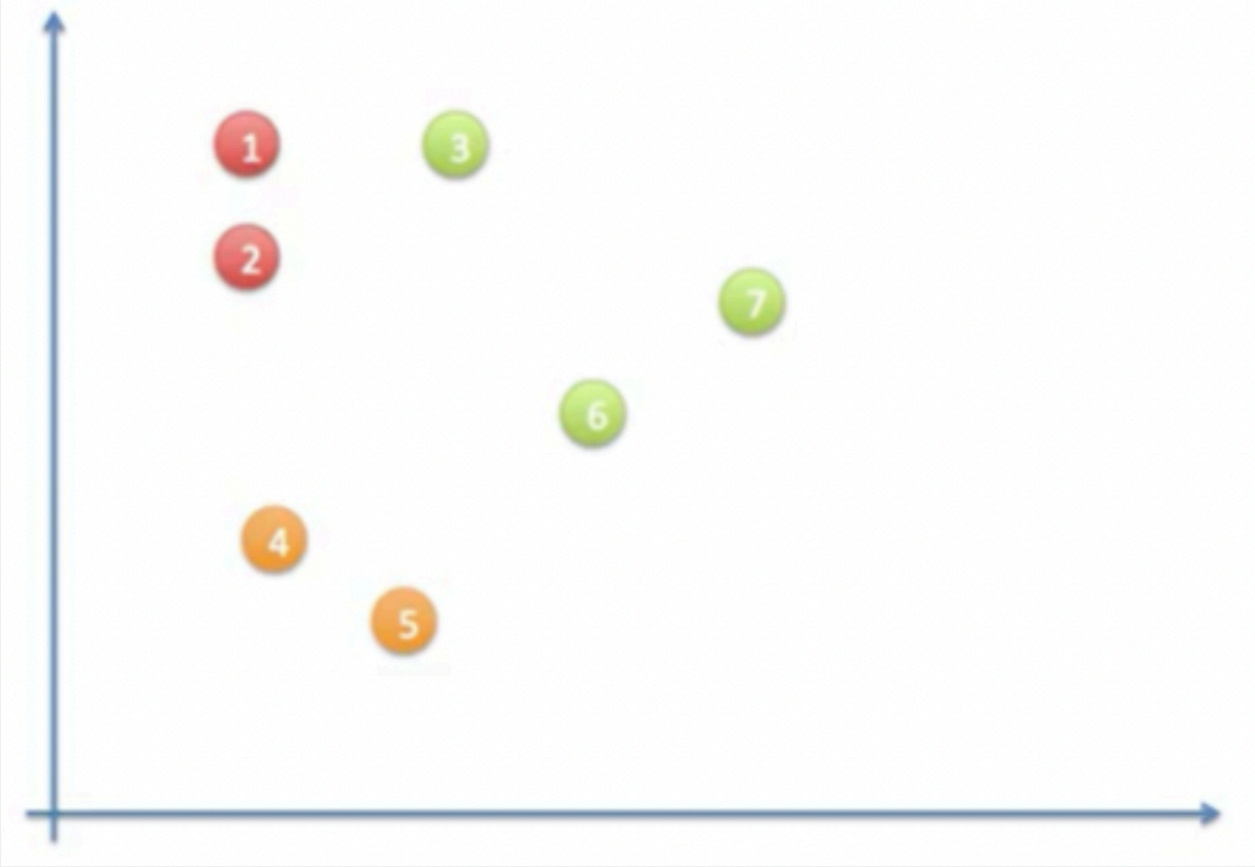
4. 클러스터 할당 -> 클러스터 중심 이동 : 반복



03 K-Means Clustering

-K-Means Clustering Process

5. 최종 결과



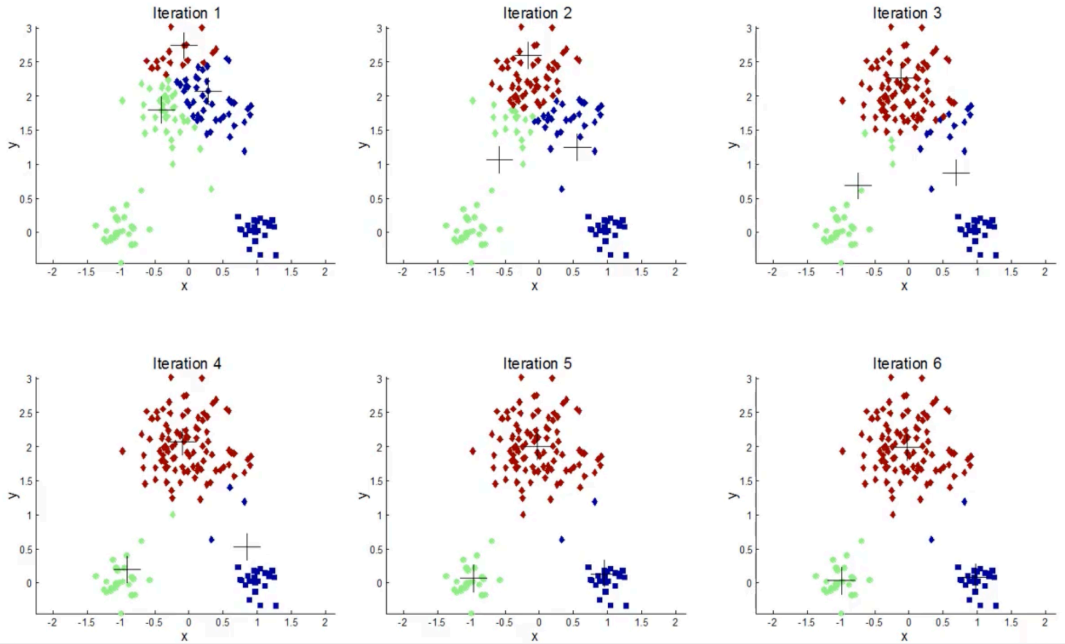
03 K-Means Clustering

-Effect of initial centroids

초기값에 따른 다른 클러스터링

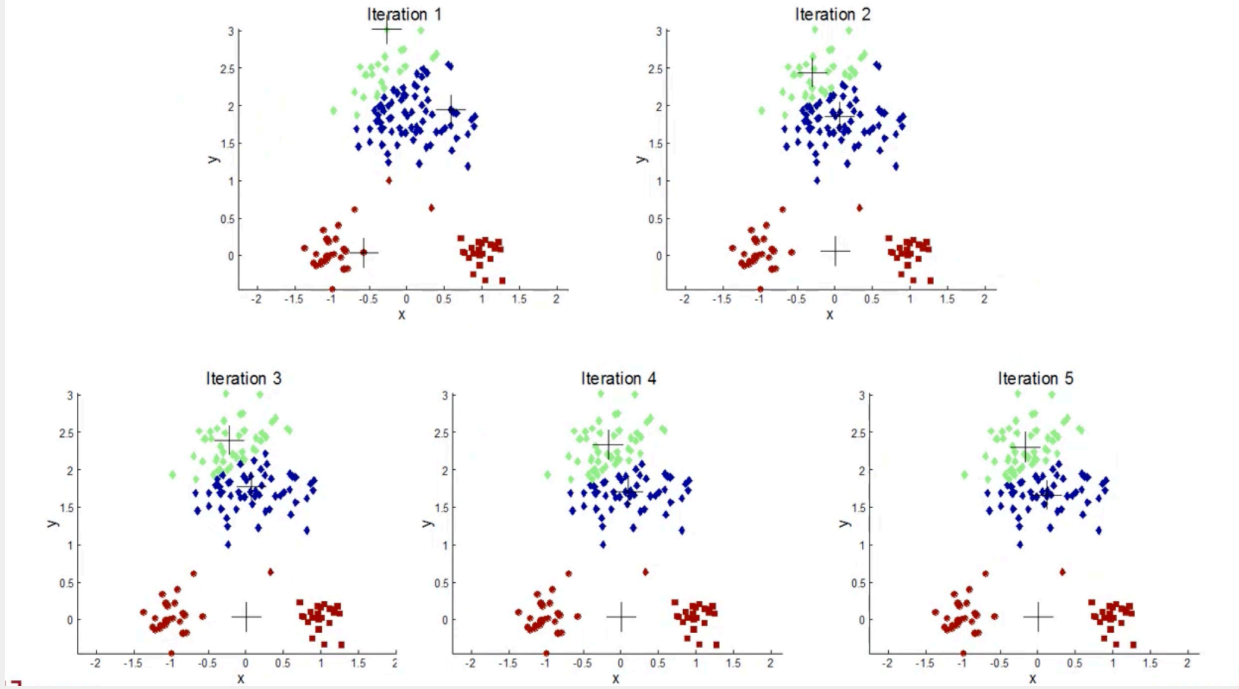
-이상적인 클러스터링

- Effect of initial centroids
  - ✓ Desirable centroid selection



-이상적이지 않은 클러스터링

✓ Undesirable centroid selection



03 K-Means Clustering

-Remedies for initial centroid selection

- Some remedies for initial centroid selection
  - ✓ Multiple runs
  - ✓ Sample and use hierarchical clustering to determine initial centroids
  - ✓ Preprocessing & Postprocessing

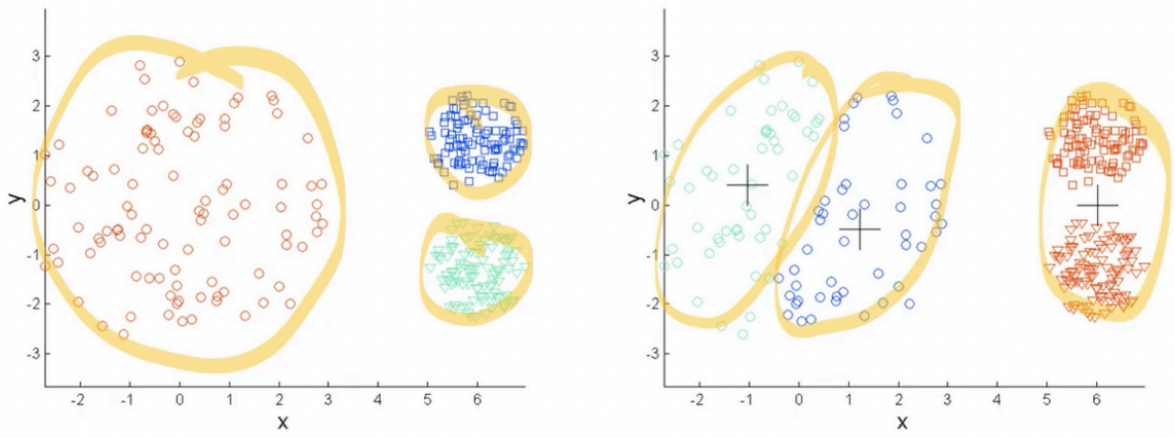
03 K-Means Clustering

-Limitations of K-Means Clustering

✓ K-Means Clustering의 한계점

- 밀도가 다를 경우

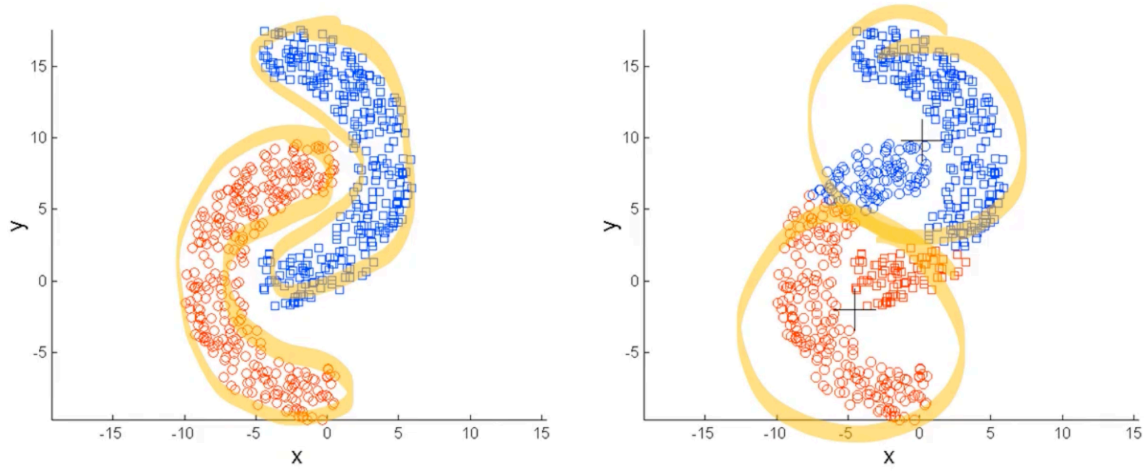
- Limitations of K-Means Clustering
  - ✓ Cannot cope with different densities



- 구의 형태가 아닐 경우

Limitations of K-Means Clustering

✓ Cannot cope with non-globular shapes



CONTENTS

04. DBSCAN

- [x] Overview
- [x] EPSILON
- [x] Border Point
- [x] Definition of DBSCAN

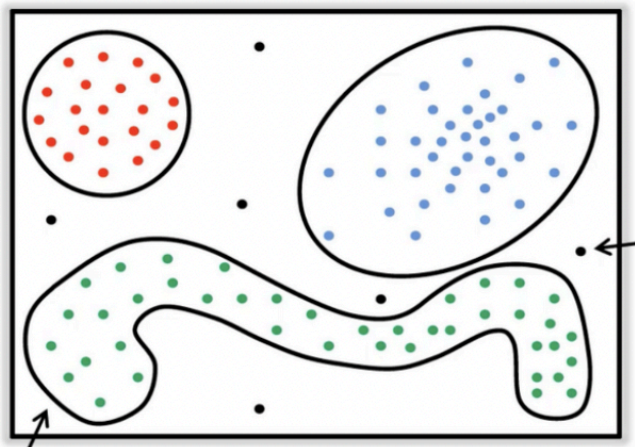
04 DBSCAN

-Overview

Density-based Clustering

Ester et al. (1996)

- Density-based clustering
  - ✓ Conduct a clustering by considering the density of data points
    - Can find an arbitrary shape of cluster
    - Can remove noise from clustering result



✓ 밀도를 정확히 파악

✓ 임의의 모양의 군집을 찾아낼 수 있음

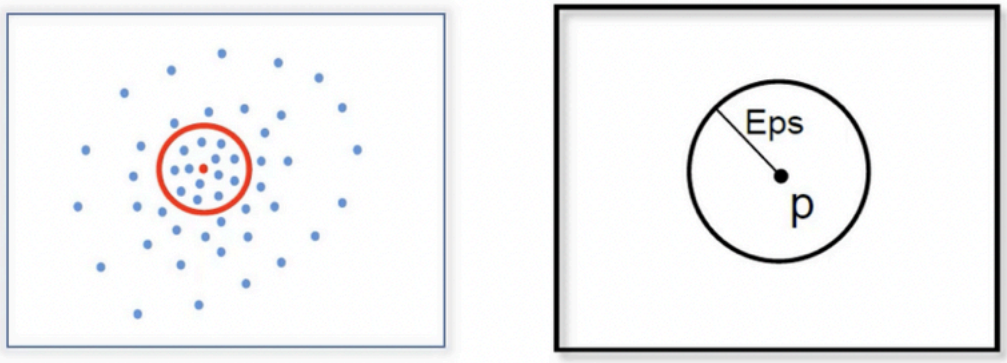
✓ 어떤 cluster에도 할당되지 않는 객체(noise)들도 존재할 수 있음

04 DBSCAN

-EPSILON

- DBSCAN
  - ✓ Definition 1: ε-neighborhood of a point
    - The ε-neighborhood of a point, denoted by $N_\varepsilon(p)$, is defined by

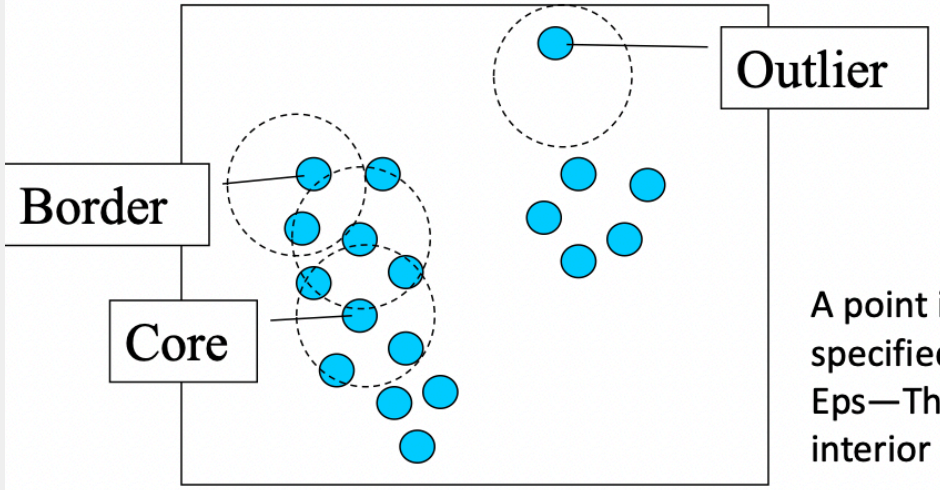
$$N_\epsilon(p) = \{q \in D \mid dist(p, q) \leq \epsilon\}$$



  - ✓ Naïve Approach: require for each point in a cluster that there are at least a minimum number (MinPts) of points in an ε-neighborhood of that point

04 DBSCAN

-Border Point



Given ε and *MinPts*, categorize the objects into three exclusive groups.

A point is a core point if it has more than a specified number of points (MinPts) within Eps—These are points that are at the interior of a cluster.

$\varepsilon = 1\text{unit}, \text{MinPts} = 5$

A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point.

A noise point is any point that is not a core point nor a border point.

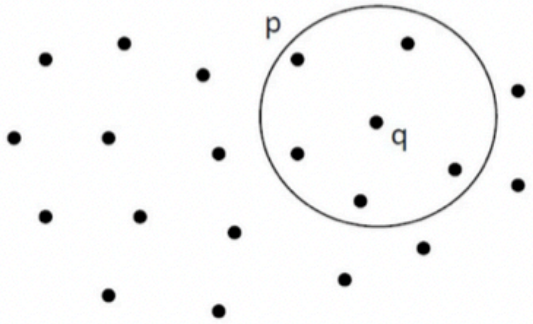
04 DBSCAN

-Definition of DBSCAN

- DBSCAN
  - ✓ Definition 2: directly density-reachable
    - A point p is directly density-reachable from a point q with regard to the parameters ε and MinPts, if

$$1)\ \ p \in N_\epsilon(q)\ \ (reachability)$$

$$2)\ \ |N_\epsilon(q)| \geq MinPts\ \ (core\ point\ condition)$$



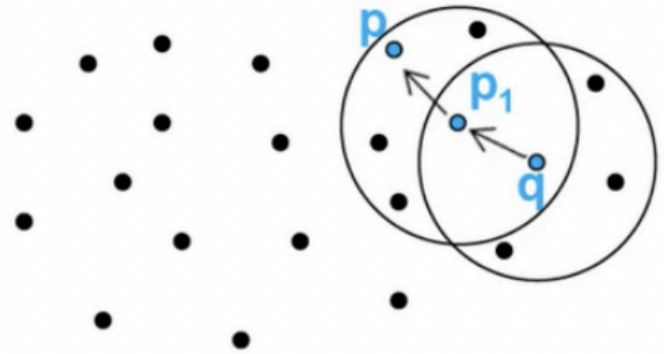
MinPts = 5

$| N_{Eps}(q) | = 6 \geq 5 = MinPts$ (core point condition)

04 DBSCAN

-Definition of DBSCAN

- DBSCAN
  - ✓ Definition 3: density-reachable
    - A point p is density-reachable from a point q with regard to the parameters ε and MinPts, if there is a chain of points $p_1, p_2, \ldots, p_s$ with $p_1 = q$ and $p_s = p$ such that p_{i+1} is directly density-reachable from p_i for all $1 < 1 < s-1$



MinPts = 5

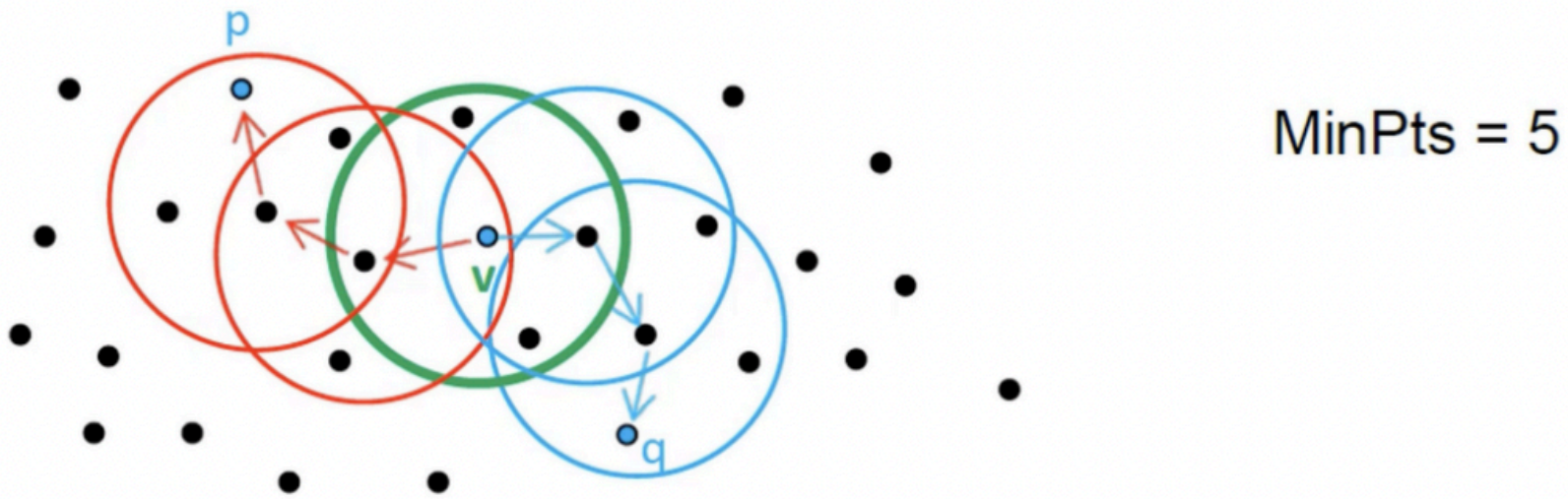
$| N_{Eps}(q) | = 5 = MinPts$ (core point condition)

$| N_{Eps}(p_1) | = 6 \geq 5 = MinPts$ (core point condition)

04 DBSCAN

-Definition of DBSCAN

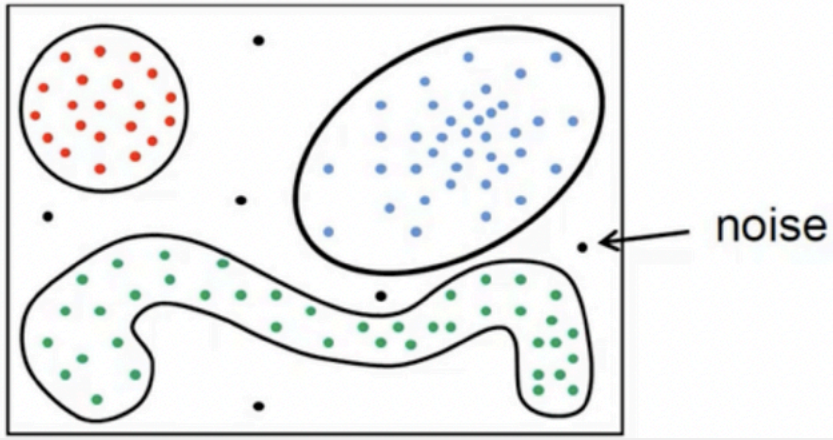
- DBSCAN
  - ✓ Definition 4: density-connected
    - A point p is density-connected to a point q with regard to the parameters ε and MinPts, if there is a point v such that both p and q are density-reachable from v



04 DBSCAN

-Definition of DBSCAN

- DBSCAN
  - ✓ Definition 5: Cluster
    - A cluster with regard to the parameters ε and MinPts is a non-empty subset C of the database D with
      - (1) For all p, q ∈ D: If p ∈ C and q is density-reachable from p with regard to the parameters ε and MinPts, then q ∈ C (Maximality)
      - (2) For all p, q ∈ C: The point p is density-connected to q with regard to the parameters ε and MinPts (Connectivity)



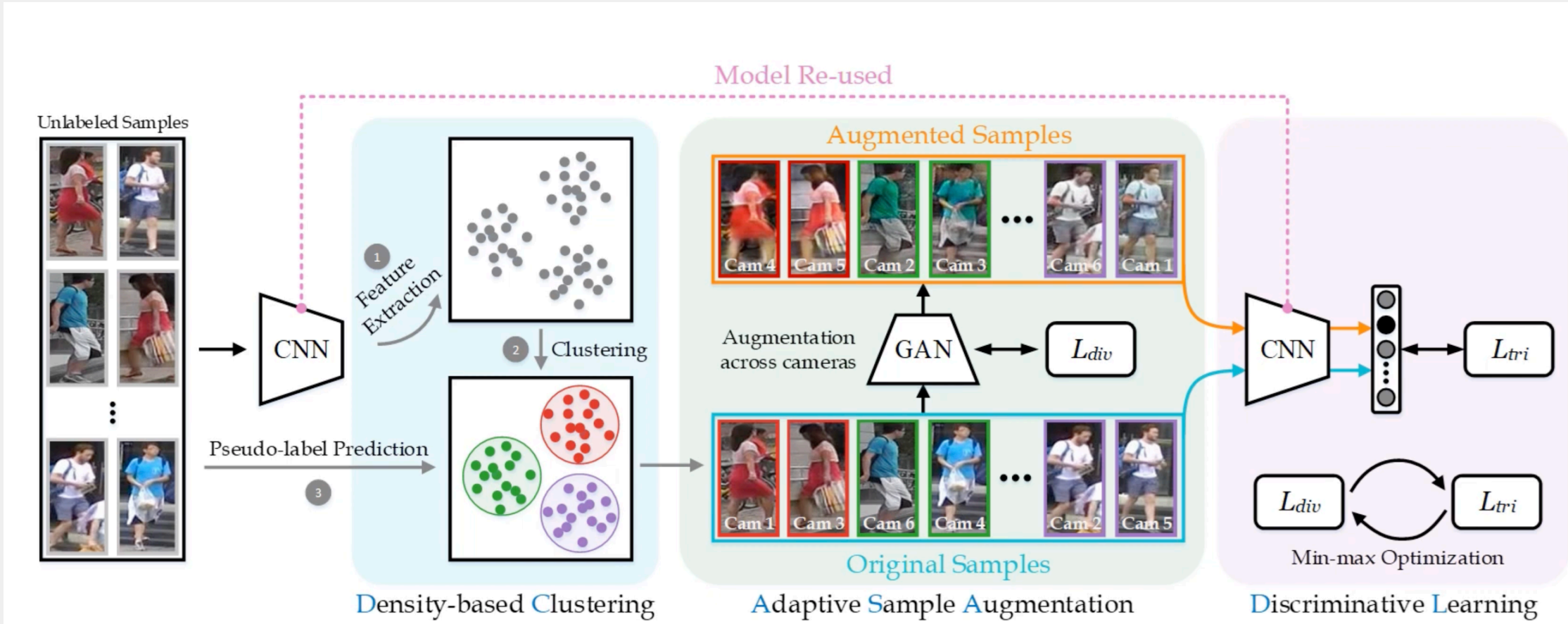
CONTENTS

05. Clustering with data augmentation

☑ Project

05 Clustering with data argumentation

-Project



05 Clustering with data argumentation

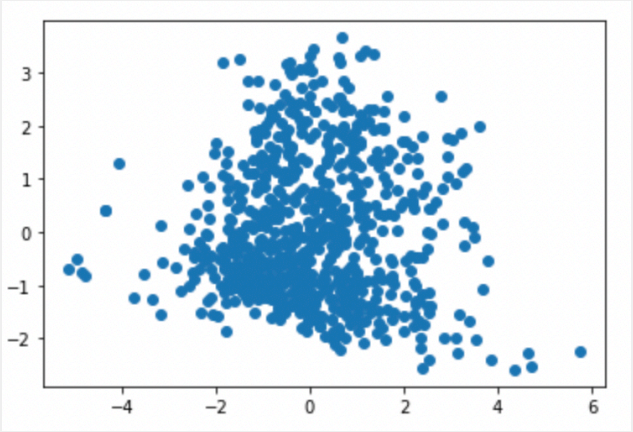
-Project

우리의 프로젝트 목표 : 수집한 데이터가 적을 때 clustering을 통해 데이터를 추가해보자.

데이터 추가방식

1. PCA를 통해 2차원으로 데이터를 축소
2. PCA한 데이터 -> clustering
3. cluster의 중심점과 중심점에서 거리가 가장 먼 점을 반지름으로 설정하여 원을 그림
4. 그린 원을 기준으로 데이터를 추가

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0



05 Clustering with data argumentation

-Project

1. 데이터 수집

```
import pandas as pd
df = pd.read_csv("diabetes.csv")
```

2. 데이터 -> 2차원으로 변형

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler() #PCA를 위한 scaling
X = df.drop(["Outcome"], axis = 1)
y = df.Outcome

X_scaled = scaler.fit_transform(X)

pca = PCA(n_components = 2) #거리 공식 사용 및 시각화에 용의하게 하기 위해 pca 적용
X_pca = pca.fit_transform(X_scaled)

ndf= pd.DataFrame(X_pca)
ndf["Outcome"] = y

ndf.columns=['x','y', 'Outcome']
```

	x	y	Outcome
0	1.068503	1.234895	1
1	-1.121683	-0.733852	0
2	-0.396477	1.595876	1
3	-1.115781	-1.271241	0
4	2.359334	-2.184819	1
...	...	...	...
763	1.562085	1.923150	0
764	-0.100405	-0.614181	0
765	-0.283475	0.097065	0
766	-1.060324	0.837062	1
767	-0.839892	-1.151755	0

05 Clustering with data argumentation

-Project

3. 데이터 추가하지 않은 경우의 RandomForestClassifier의 성능

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()

X = ndf.drop(["Outcome"], axis = 1)
y = ndf.Outcome

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

model.fit(X_train, y_train)
pred = model.predict(X_test)

from sklearn.metrics import accuracy_score

print(accuracy_score(pred, y_test))
```

```
0.6428571428571429
```

05 Clustering with data argumentation

-Project

4. cluster를 이용하여 새로운 데이터프레임 구축

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2,init='k-means++',n_init=10,max_iter=300,random_state=0)
temp = kmeans.fit(X)
ndf["cluster"] = -1
ndf["cluster"] = temp.labels_
cluster0_center, cluster1_center = temp.cluster_centers_[0], temp.cluster_centers_[1]
```

```
temp.labels_
```

```
array([0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0,
```

05 Clustering with data argumentation

-Project

4. cluster를 이용하여 새로운 데이터프레임 구축

```
df_zero = ndf[ndf.cluster == 0]
df_one = ndf[ndf.cluster == 1]
df_zero["dist"] = 10000000
df_one["dist"] = 10000000
```

df_zero

	x	y	Outcome	cluster	dist
0	1.068503	1.234895	1	0	0.584855
2	-0.396477	1.595876	1	0	0.932334
8	3.297925	-0.242155	1	0	3.248557
9	-1.839850	3.206661	1	0	2.945939
10	-0.532554	0.647607	0	0	1.326168
...	...	...	...	...	...
759	1.085151	3.331802	1	0	1.964723
761	1.675967	1.565718	1	0	1.157686
762	-2.014177	1.499662	0	0	2.538879
763	1.562085	1.923150	0	0	1.141126
766	-1.060324	0.837062	1	0	1.698527

```
for i in range(len(df_zero)):
    df_zero["dist"].iloc[i] = ((cluster0_center[0] - df_zero.iloc[i]['x'])**2 +
                               (cluster0_center[1] - df_zero.iloc[i]['y'])**2)**(1/2)
for i in range(len(df_one)):
    df_one["dist"].iloc[i] = ((cluster1_center[0] - df_one.iloc[i]['x'])**2 +
                              (cluster1_center[1] - df_one.iloc[i]['y'])**2)**(1/2)
```

05 Clustering with data argumentation

-Project

5. 새로운 데이터프레임을 바탕으로 클러스터의 중심으로부터 가장 먼 거리를 가지는 점을 구하기

```
sort_dfzero=df_zero.sort_values("dist", ascending = False)
zero_maxdist=sort_dfzero['dist'].iloc[0]
```

```
sort_dfone=df_one.sort_values("dist", ascending = False)
one_maxdist=sort_dfone['dist'].iloc[0]
```

sort_dfzero

	x	y	Outcome	cluster	dist
13	3.787107	-0.525323	1	0	3.813661
487	3.486637	-0.083464	0	0	3.335281
8	3.297925	-0.242155	1	0	3.248557
579	3.470046	0.083450	1	0	3.246906
43	3.597245	1.982248	1	0	3.118999
...	...	...	...	...	...
330	0.358473	1.367674	0	0	0.184539
30	0.598062	1.589228	0	0	0.158617
254	0.423444	1.563911	1	0	0.152927
41	0.395740	1.375969	0	0	0.147694
25	0.447031	1.445316	1	0	0.077243

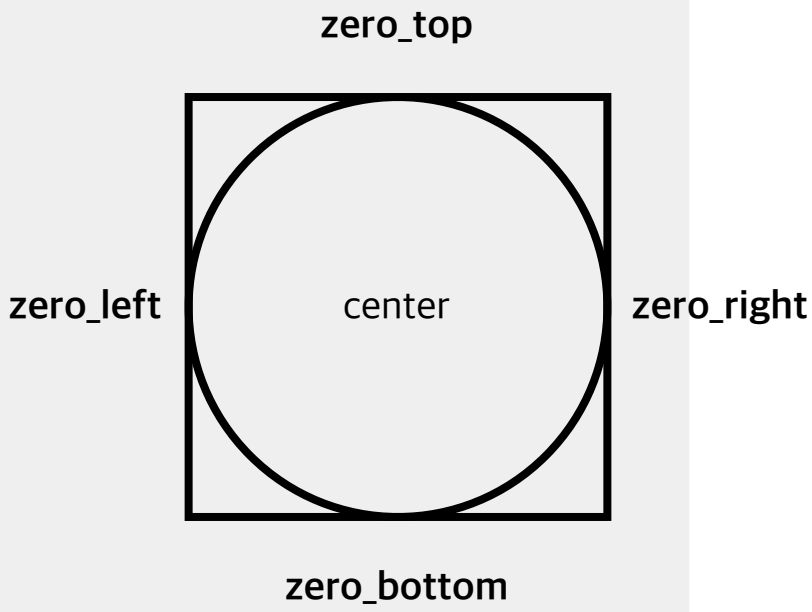
05 Clustering with data argumentation

-Project

6. Data Argumentation

```
zero_top = cluster0_center[1] + zero_maxdist
zero_bot = cluster0_center[1] - zero_maxdist
zero_right = cluster0_center[0] + zero_maxdist
zero_left = cluster0_center[0] - zero_maxdist

import random
lst_zero_x, lst_zero_y = [], []
while True:
    x = random.uniform(zero_left, zero_right)
    y = random.uniform(zero_bot, zero_top)
    if ((x - cluster0_center[0])**2 + (y - cluster0_center[1])**2)**(1/2) > zero_maxdist:
        continue
    else:
        lst_zero_x.append(x)
        lst_zero_y.append(y)
    if len(lst_zero_x) == 1200 :
        break
```




```
temp1 = pd.DataFrame(lst_zero_x)
temp2 = pd.DataFrame(lst_zero_y)
temp = pd.concat([temp1, temp2], axis = 1)
temp.columns = ['x', 'y']
temp["Outcome"] = 0
temp["cluster"] = 0
```

	x	y	Outcome	cluster
0	-2.292127	2.753694	0	0
1	0.898441	1.955625	0	0
2	-1.632569	0.086316	0	0
3	0.521336	-0.419069	0	0
4	1.381007	-1.470802	0	0
...	...	...	...	...
1195	2.151957	2.874976	0	0
1196	2.252287	-0.613785	0	0
1197	2.129576	2.978655	0	0
1198	-1.395972	0.525644	0	0
1199	2.859190	3.091407	0	0

05 Clustering with data argumentation

-Project

6. Data Argumentation

```
one_top = cluster1_center[1] + one_maxdist
one_bot = cluster1_center[1] - one_maxdist
one_right = cluster1_center[0] + one_maxdist
one_left = cluster1_center[0] - one_maxdist

import random
lst_one_x, lst_one_y = [], []
while True:
    x = random.uniform(one_left, one_right)
    y = random.uniform(one_bot, one_top)
    if ((x - cluster1_center[0])**2 + (y - cluster1_center[1])**2)**(1/2) > one_maxdist:
        continue
    else:
        lst_one_x.append(x)
        lst_one_y.append(y)
    if len(lst_one_x) == 1200:
        break
```

```
temp1 = pd.DataFrame(lst_one_x)
temp2 = pd.DataFrame(lst_one_y)
ntemp = pd.concat([temp1, temp2], axis = 1)
ntemp.columns = ['x', 'y']
ntemp["Outcome"] = 1
ntemp["cluster"] = 1
```

05 Clustering with data argumentation

-Project

```
temp1 = pd.DataFrame(lst_one_x)
temp2 = pd.DataFrame(lst_one_y)
ntemp = pd.concat([temp1, temp2], axis = 1)
ntemp.columns = ['x', 'y']
ntemp["Outcome"] = 1
ntemp["cluster"] = 1
```

```
nntemp = pd.concat([temp, ntemp], axis = 0)
nntemp.reset_index(inplace = True)
nntemp.drop(['index'], axis = 1, inplace = True)
```

```
df = pd.concat([ndf, nntemp], axis = 0)
df.reset_index(inplace = True)
df.drop(["index"], axis = 1, inplace = True)
```

```
X = df.drop(["Outcome", "cluster"], axis = 1)
y = df["Outcome"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42, stratify = y)
model = RandomForestClassifier()
model.fit(X_train, y_train)
pred = model.predict(X_test)
```

```
acc = accuracy_score(y_test, pred)
```

```
print(acc)
```

```
0.7208201892744479
```

05 Clustering with data argumentation

-Project

-의의

✓ Data Argumentation을 clustering으로 해본다는 창의적인 아이디어 제공
✓ Clustering을 보는 관점을 다르게 한다.
✓ 데이터가 밀집되어 있어 clustering 후 원을 그렸을 때 클러스터 간의 교집합이 크므로 정확도가 크게 개선되지 않을 것이라 생각했지만, accuracy가 8% 증가했습니다.

-한계

✓ 계산 편의를 위해 2차원으로 차원 축소한 부분
✓ K-Means clustering 외의 다른 클러스터링 모델을 다양하게 사용했으면 더 좋았을 것 같음.

감사합니다