Basics of hierarchical clustering

CLUSTER ANALYSIS IN PYTHON

Shaumik Daityari
Business Analyst

datacamp

Creating a distance matrix using linkage

계층적 clustering을 수행하는데 중요한 단계는 각 단계에서 거리 행렬을 계산하는 것

```
scipy.cluster.hierarchy.linkage(observations,
                                method='single',
                                metric='euclidean',
                                optimal_ordering=False
)
```

관측 값

이 프로세스는 거리를 계산
N cluster에서 1 cluster로 이동할 때
cluster 사이에서 N은 point의 수

두 cluster 간의 근접성을 계산하는 방법을 알고리즘에 알려줌

metric : 두 물체 사이의 거리를 결정하는 기능
유클리드 : 2D 평면의 두 점 사이의 직선 거리

연결 행렬의 순서를 변경하는 선택적 인수

- `method` : how to calculate the proximity of clusters
- `metric` : distance metric
- `optimal_ordering` : order data points

Which method should use? 두번째 매개변수인 method는 각 단계에서 cluster를 분리하는 방법을 결정

- single: based on two closest objects
- complete: based on two farthest objects
- average: based on the arithmetic mean of all objects
- centroid: based on the geometric mean of all objects
- median: based on the median of all objects
- ward: based on the sum of squares

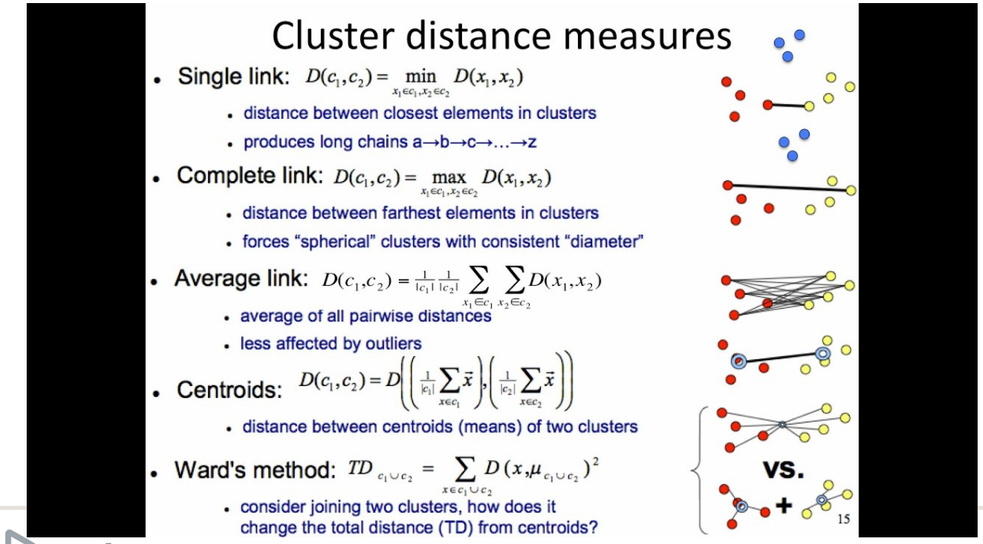
single : 가장 가까운 두 객체를 기반으로 cluster의 근접성을 결정

가장 멀리 있는 두 개의 개체를 기반으로 cluster centr의 근접성을 결정

평균 및 중심 method는 각각 산술 평균과 기하 평균을 기반으로 cluster proximities(근접) 결정

중위수 방법에서는 군집 개체의 중위수를 사용

병동법은 접합 군집의 합산 제곱에서 개별 합산 제곱을 뺀 차이를 사용하여 군집 근접도를 계산 (병동법은 중심을 향해 더 동심원화된 군집에 초점을 맞춘다)



4) Ward 연결법 (Ward's method)
군집 간 정보의 손실을 최소화하는 군집화
군집 내 편차들의 제곱합을 고려하여 군집 내 거리(within cluster distance, ESS)를 최소화.
비슷한 크기의 군집을 생성하는 경향

Create cluster labels with fcluster

거리 행렬을 만든 후에는 cluster label을 만들 수 있음

```
scipy.cluster.hierarchy.fcluster(distance_matrix,
                                 num_clusters,
                                 criterion
)
```

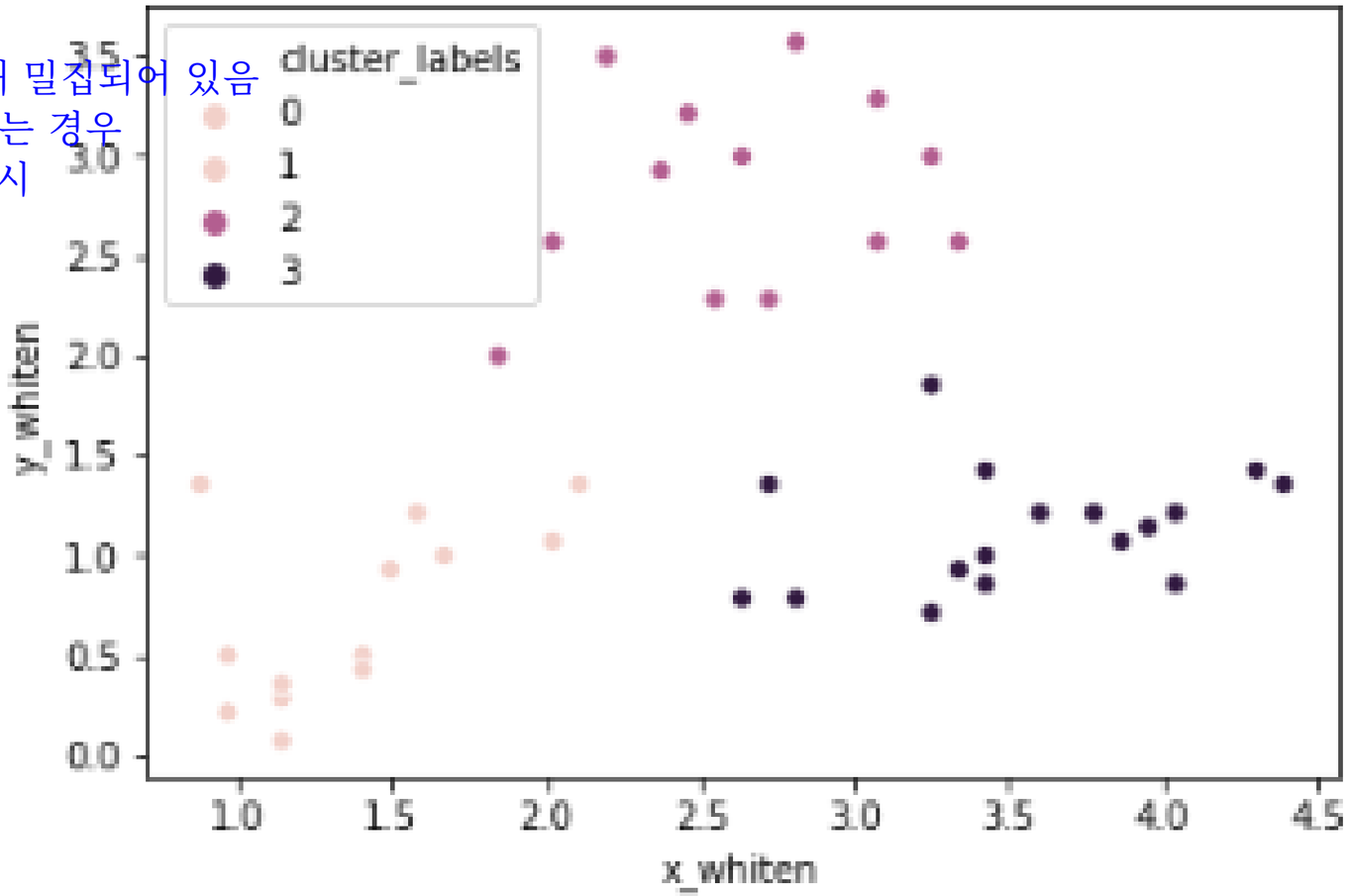
3개의 인수 : 거리 행렬, cluster의 수와 특정 임계값을 기반으로 cluster를 형성하는 기준

- `distance_matrix` : output of `linkage()` method
- `num_clusters` : number of clusters
- `criterion` : how to decide thresholds to form clusters

criterion 인수에서는 maxcluster 값을 사용

Hierarchical clustering with ward method

ward method를 사용한 결과
cluster는 일반적으로 중심을 향해 밀집되어 있음
모든 seaborn plot에는 개체가 없는 경우
label 0이 있는 추가 cluster가 표시



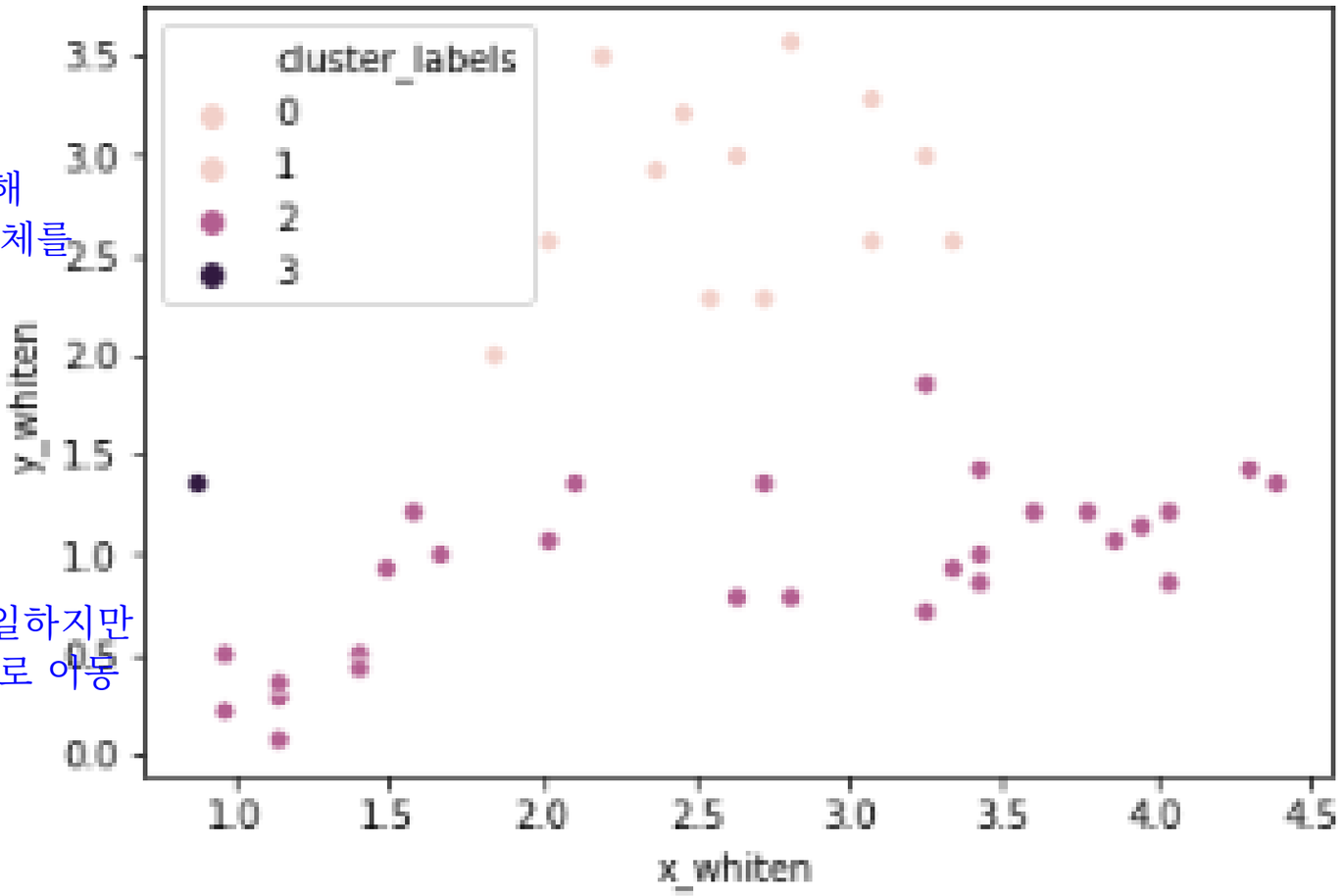
Hierarchical clustering with single method

single method 사용

cluster 간 근접성을 결정하기 위해
cluster 사이에 가장 가까운 두 객체를
사용한 single mothod

이 방법을 통해 clustering을
수행할 때 형성되는 cluster는
더 많이 분산됨

1로 표시된 맨 위 군집은 거의 동일하지만
군집 3의 대부분의 개체가 군집 2로 이동

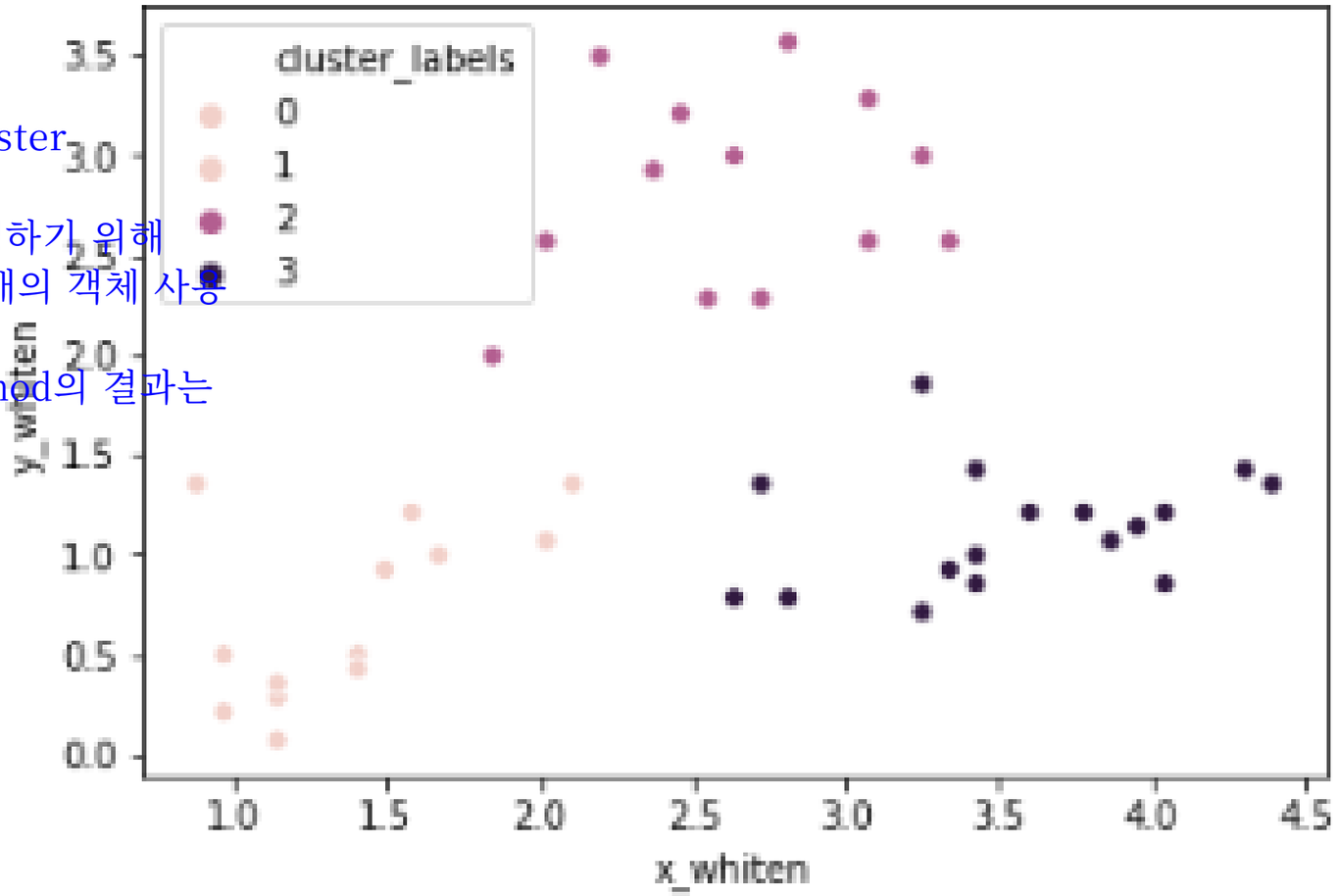


Hierarchical clustering with complete method

complete method으로 형성된 cluster

이 방법은 cluster간 근접성을 결정하기 위해
cluster 중에서 가장 멀리 있는 두개의 객체 사용

그러나 공료롭게도 complete method의 결과는
ward method와 유사



Final thoughts on selecting a method

- No one right method for all
- Need to carefully understand the distribution of data

이 수업을 완료하기 전에 몇 가지 생각
1. 직면한 모든 문제에 적용할 수 있는 올바른 방법은 없음
2. data를 주의 깊게 연구해야 함

Let's try some exercises

CLUSTER ANALYSIS IN PYTHON

Visualize clusters

CLUSTER ANALYSIS IN PYTHON

Shaumik Daityari
Business Analyst

datacamp

Why visualize clusters?

- Try to make sense of the clusters formed
- An additional step in validation of clusters
- Spot trends in data

cluster를 시각화해야 하는 이유?

- 단순히 cluster center를 보는 것이 아니라 시각적으로 분석함으로써 어떤 알고리즘에 의해 형성되는 cluster를 빠르게 이해할 수 있음
- 형성된 cluster를 검증하기 위해 추가 단계가 될 수 있음
- 또한 데이터를 시각적으로 살펴봄으로써 데이터의 추세를 발견할 수 있음

An introduction to seaborn

- `seaborn` : a Python data visualization library based on `matplotlib`
- Has better, easily modifiable aesthetics than matplotlib! 쉽고 직관적으로 수정할 수 있는 더 나은 기본 ploting 테마를 제공
- Contains functions that make data visualization tasks easy in the context of data analytics
- Use case for clustering: `hue` parameter for plots seaborn은 scatter에서 cluster label을 시각화할 때 서로 다른 색상을 사용하여 cluster를 구분할 수 있는 인수를 제공

Visualize clusters with matplotlib

```
from matplotlib import pyplot as plt
```

```
df = pd.DataFrame({'x': [2, 3, 5, 6, 2],
                   'y': [1, 1, 5, 5, 2],
                   'labels': ['A', 'A', 'B', 'B', 'A']})
colors = {'A':'red', 'B':'blue'}
df.plot.scatter(x='x',
                y='y',
                c=df['labels'].apply(lambda x: colors[x]))
plt.show()
```

각 cluster를 색상에 수동으로 매핑

색상을 dictionary로 정의

lambda를 사용하여 c인수에 색상 목록을 전단

Visualize clusters with seaborn

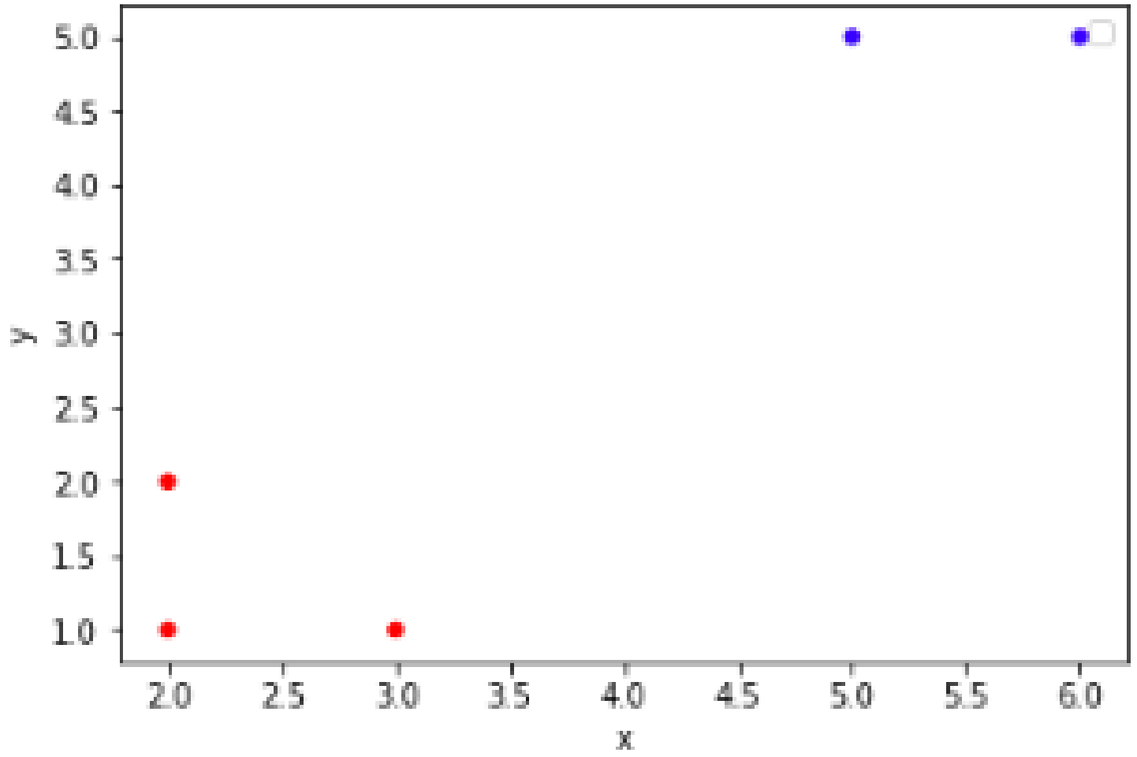
```
from matplotlib import pyplot as plt
import seaborn as sns
```

```
df = pd.DataFrame({'x': [2, 3, 5, 6, 2],
                   'y': [1, 1, 5, 5, 2],
                   'labels': ['A', 'A', 'B', 'B', 'A']})
sns.scatterplot(x='x',
                y='y',
                hue='labels',
                data=df)
plt.show()
```

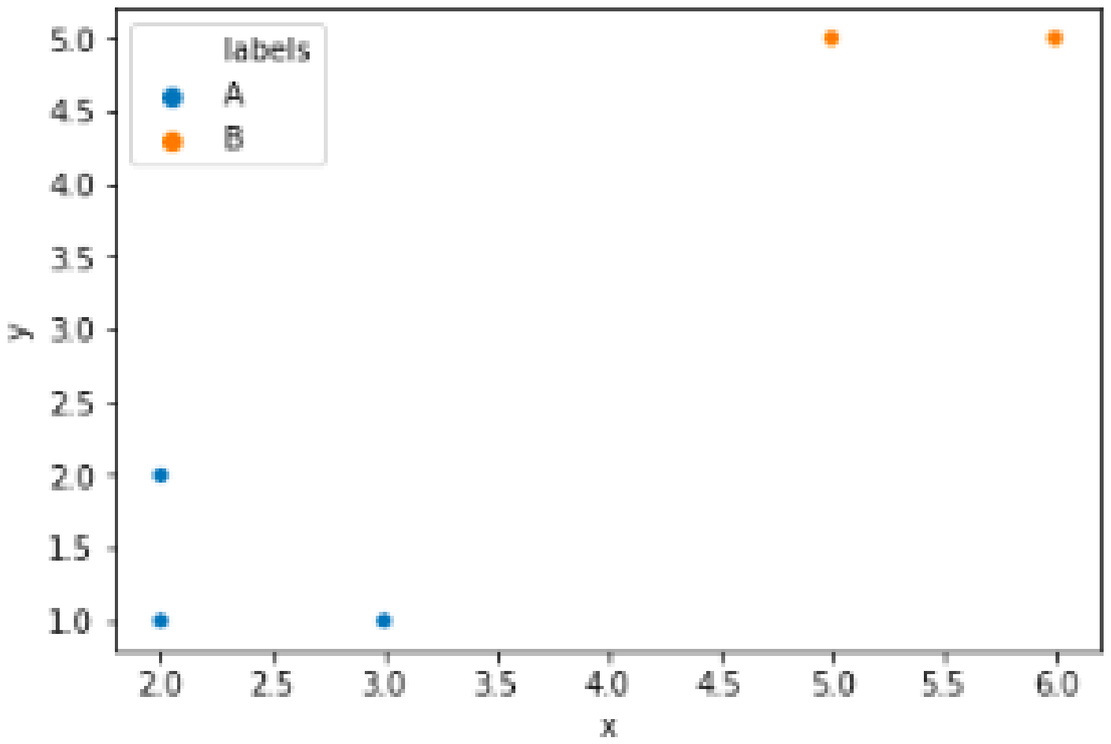
label인 cluster label의 열 이름을 전달하여 색상 wlwjd

Comparison of both methods of visualization

MATPLOTLIB PLOT



SEABORN PLOT



1. dataframe에 cluster label을 지정하면 seaborn을 사용한 구현이 더 편리해짐
2. seaborn에서 색상을 수동으로 선택할 필요가 없고 cluster 수에 관계없이 기본 팔레트를 사용

Next up: Try some visualizations

CLUSTER ANALYSIS IN PYTHON

How many clusters?

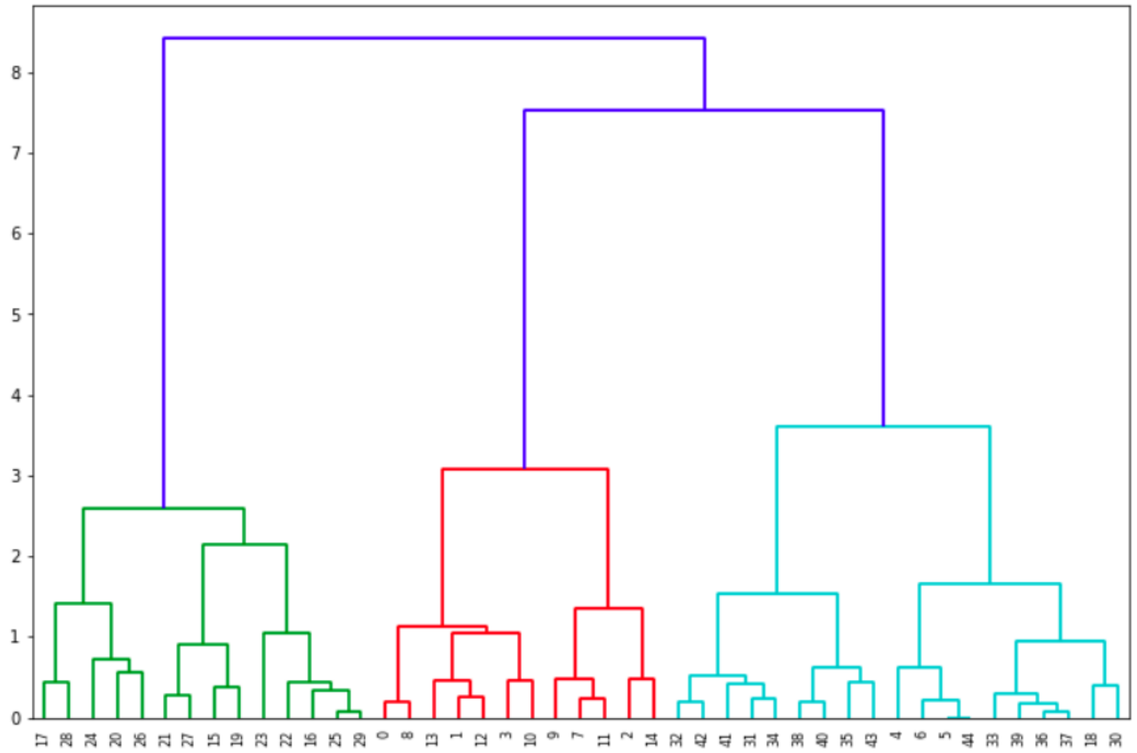
CLUSTER ANALYSIS IN PYTHON

데이터에 얼마나 많은 cluster가 있는지 결정하는 방법을 살펴봄

Shaumik Daityari
Business Analyst

Introduction to dendrograms

- Strategy till now - decide clusters on visual inspection
- Dendrograms help in showing progressions as clusters are merged
- A dendrogram is a branching diagram that demonstrates how each cluster is composed by branching out into its child nodes



- 이때까지 얼마나 많은 cluster를 형성할지 결정하기 위해 데이터 세트의 포인트 수를 그래픽으로 살펴봄
- 계층적 cluster에서 cluster 수를 결저하기 위해 덴드로그램이라는 그래픽 다이어그램을 사용할 수 있음
- 덴드로그램은 계층 클러스터링 알고리즘을 통해 진행하면서 링크 개체에서 진행률을 보여주는 분기 다이어그램

Create a dendrogram in SciPy

```
from scipy.cluster.hierarchy import dendrogram
```

```
Z = linkage(df[['x_whiten', 'y_whiten']],
            method='ward',
            metric='euclidean')
dn = dendrogram(Z)
plt.show()
```

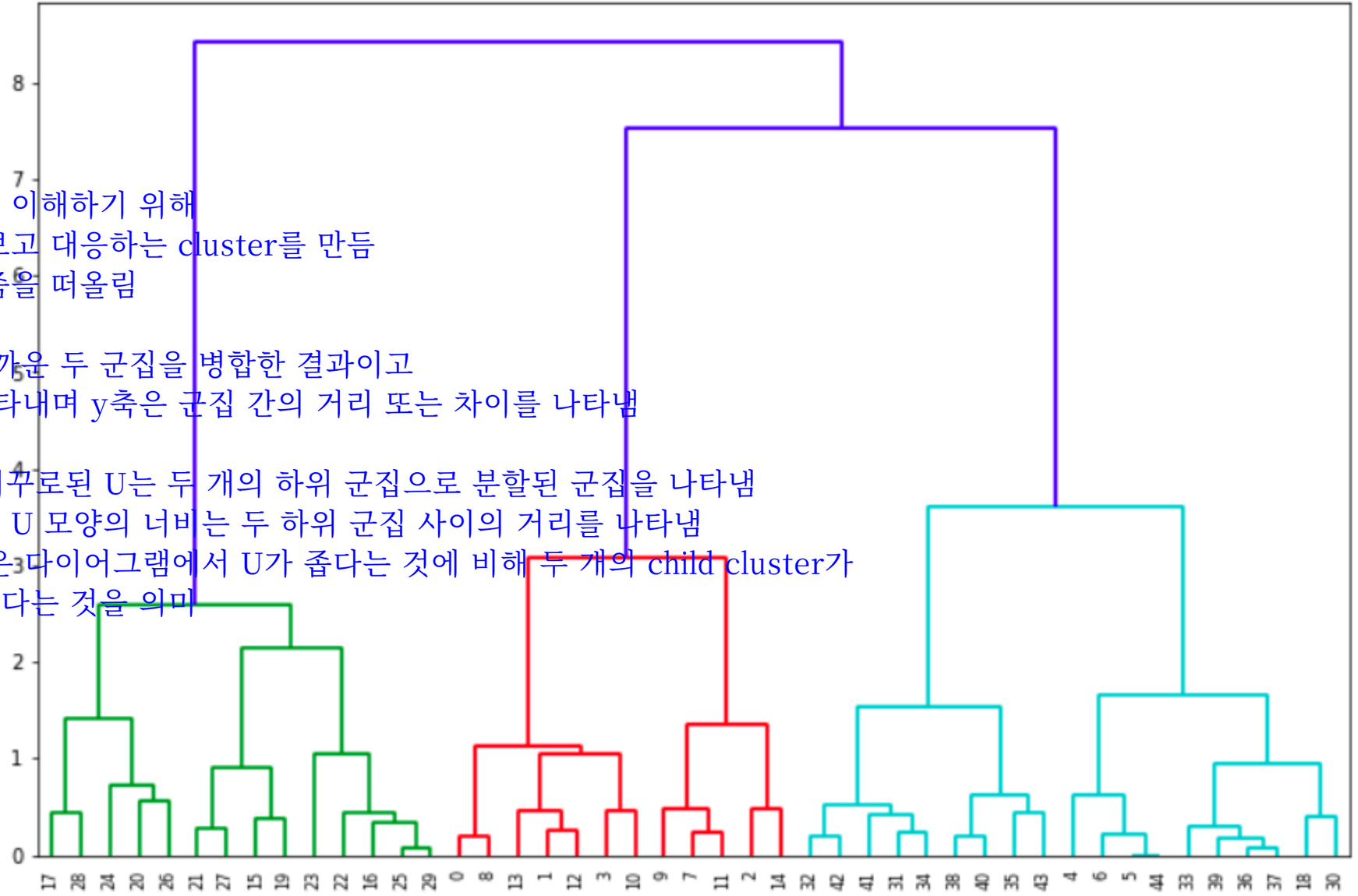
linkage를 사용하여 거리 행렬을 만듬

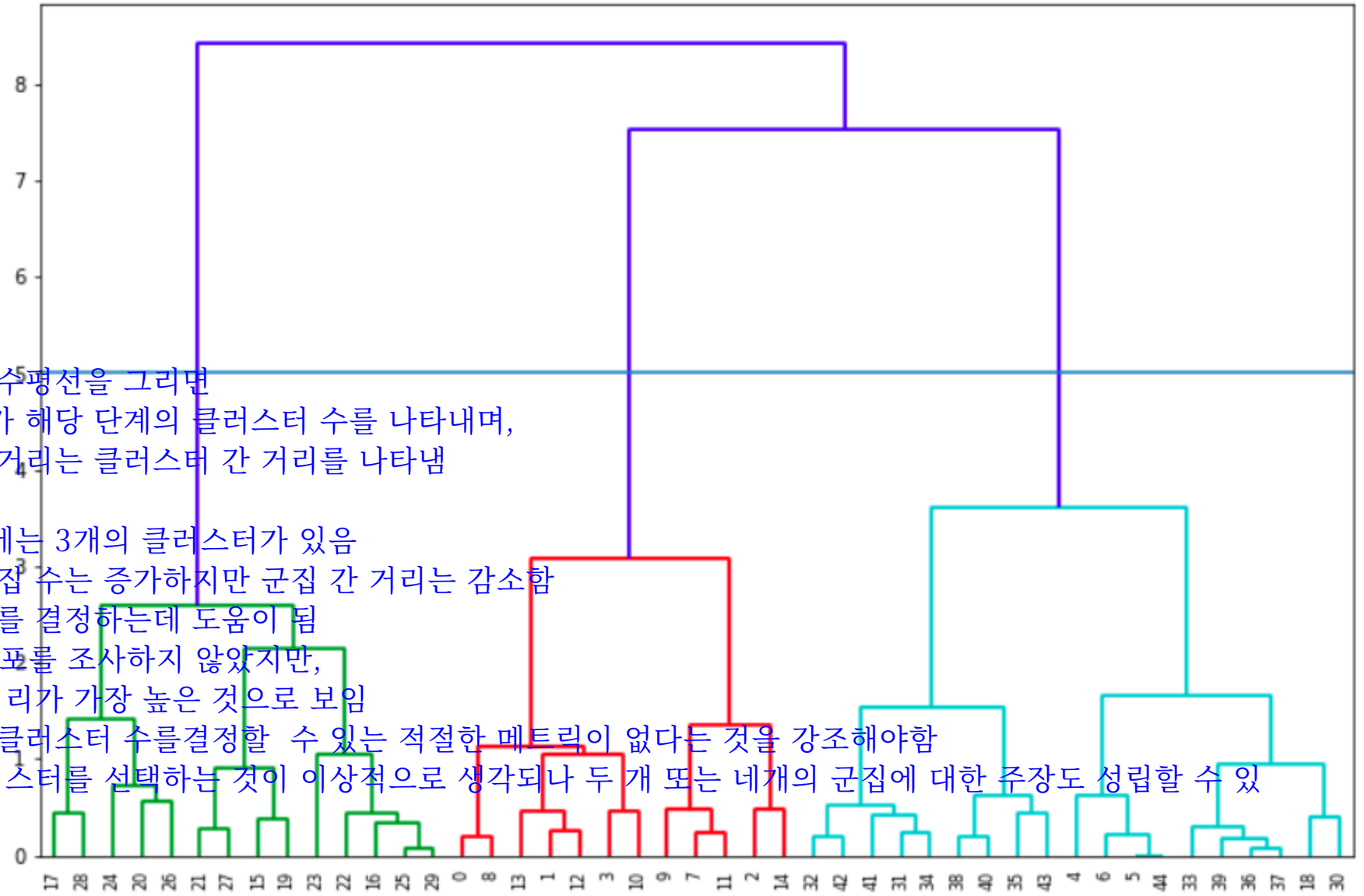
마막으로 dendrogram 방법을 사용하고 linkage 객체를 인수로 제공하여 plot을 표시

덴드로그램의 복잡성을 이해하기 위해
생성된 덴드로그램을 보고 대응하는 cluster를 만듬
계층적 군집화 알고리즘을 떠올림

이전 단계에서 가장 가까운 두 군집을 병합한 결과이고
x축은 개별 point를 나타내며 y축은 군집 간의 거리 또는 차이를 나타냄

dendrogram에서 각 거꾸로된 U는 두 개의 하위 군집으로 분할된 군집을 나타냄
그림 맨 밑에 있는 반전 U 모양의 너비는 두 하위 군집 사이의 거리를 나타냄
따라서 U가 넓다는 것은 다이어그램에서 U가 좁다는 것에 비해 두 개의 child cluster가
서로 더 멀리 떨어져 있다는 것을 의미

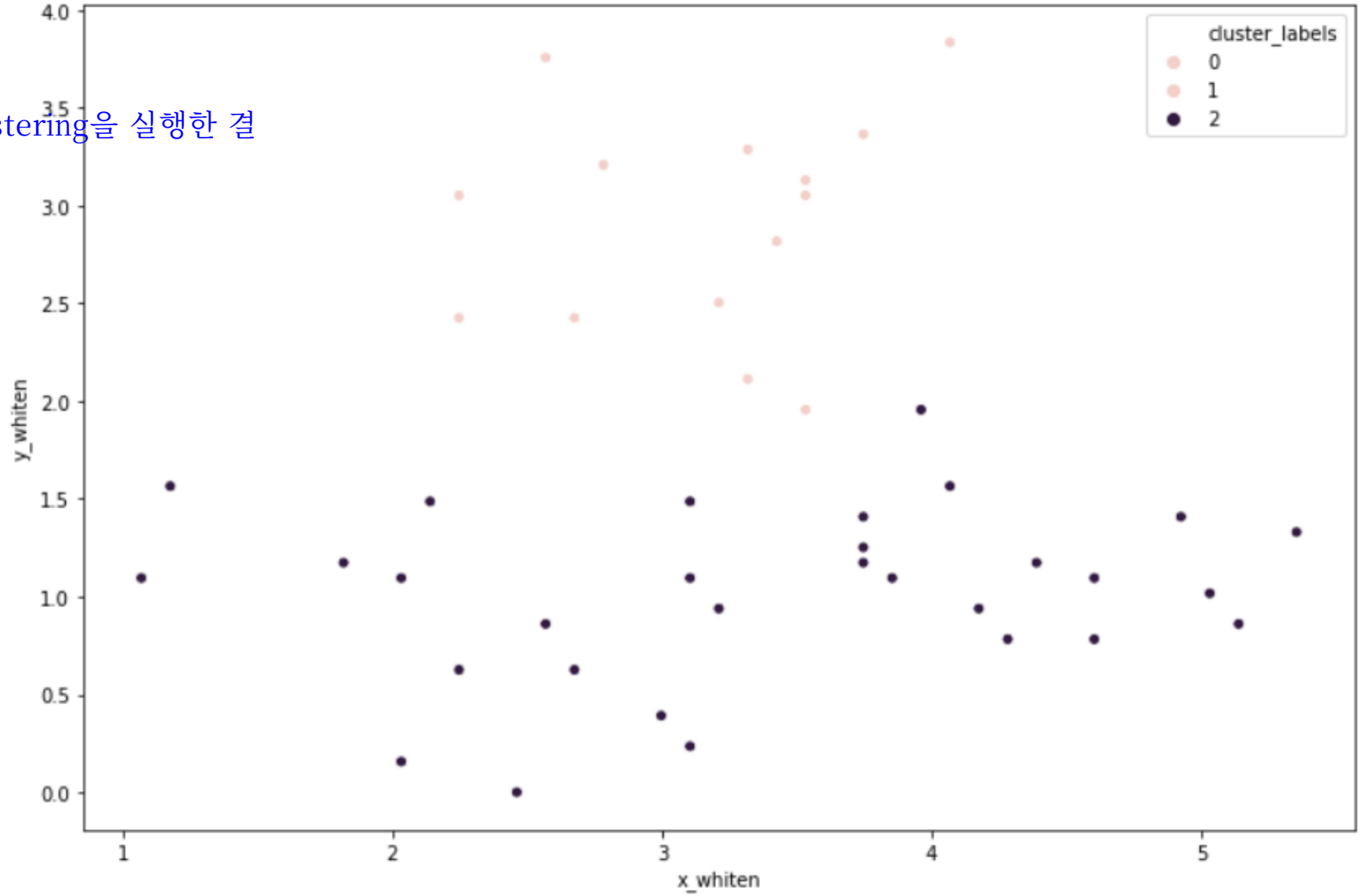




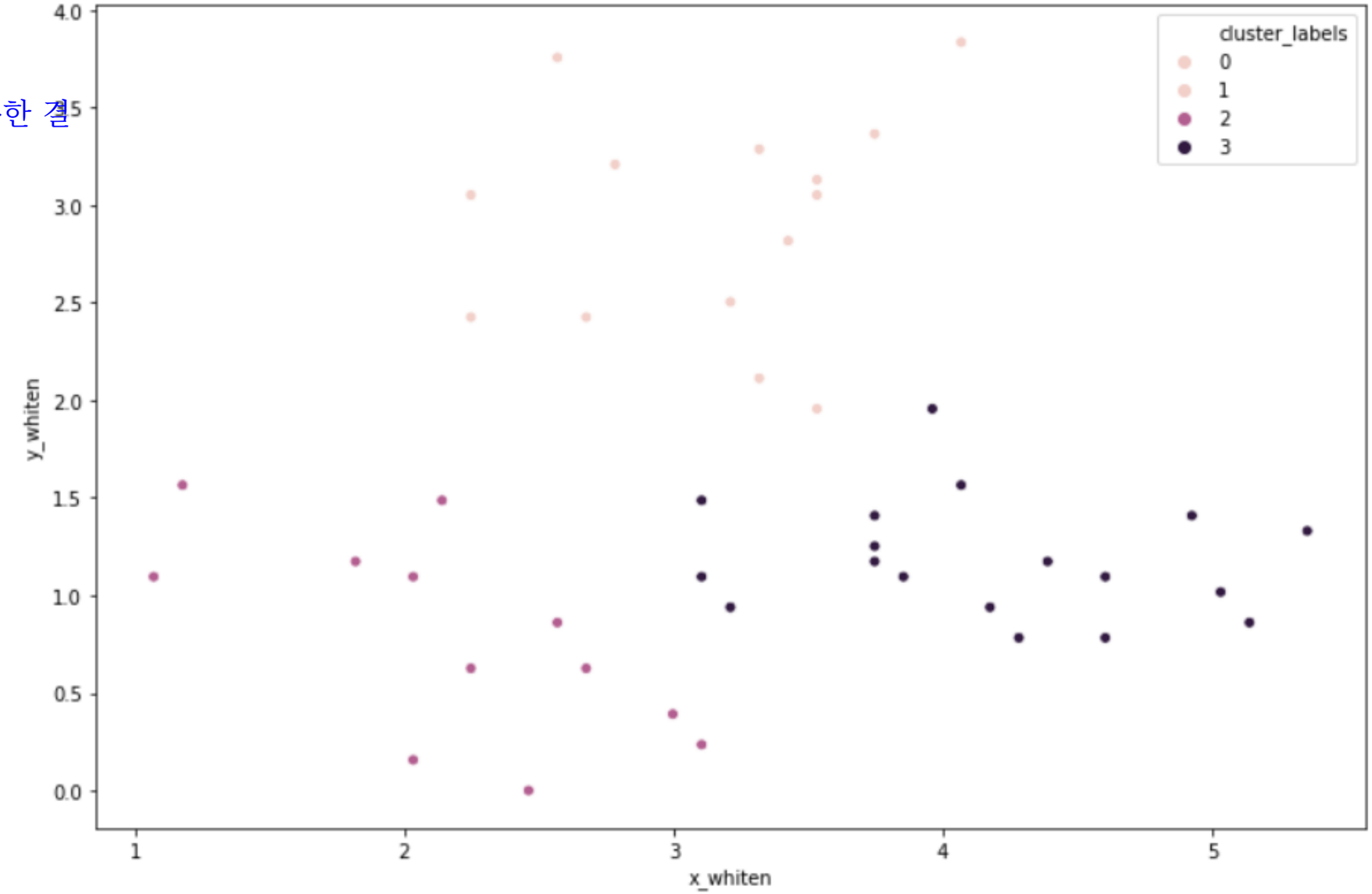
그림의 임의의 부분에 수평선을 그리면
교차하는 수직선의 수가 해당 단계의 클러스터 수를 나타내며,
이러한 수직선 사이의 거리는 클러스터 간 거리를 나타냄

그림에 그려진 수평선에는 3개의 클러스터가 있음
아래 선을 이동하면 군집 수는 증가하지만 군집 간 거리는 감소함
이 정보는 클러스터 수를 결정하는데 도움이 됨
ex) 데이터 point의 분포를 조사하지 않았지만,
상위 세 군집 사이의 거리가 가장 높은 것으로 보임
이 시점엥서 이상적인 클러스터 수를결정할 수 있는 적절한 메트릭이 없다는 것을 강조해야함
ex) 여기서 3개의 클러스터를 선택하는 것이 이상적으로 생각되나 두 개 또는 네개의 군집에 대한 주장도 성립할 수 있

2개의 클러스터로 clustering을 실행한 결

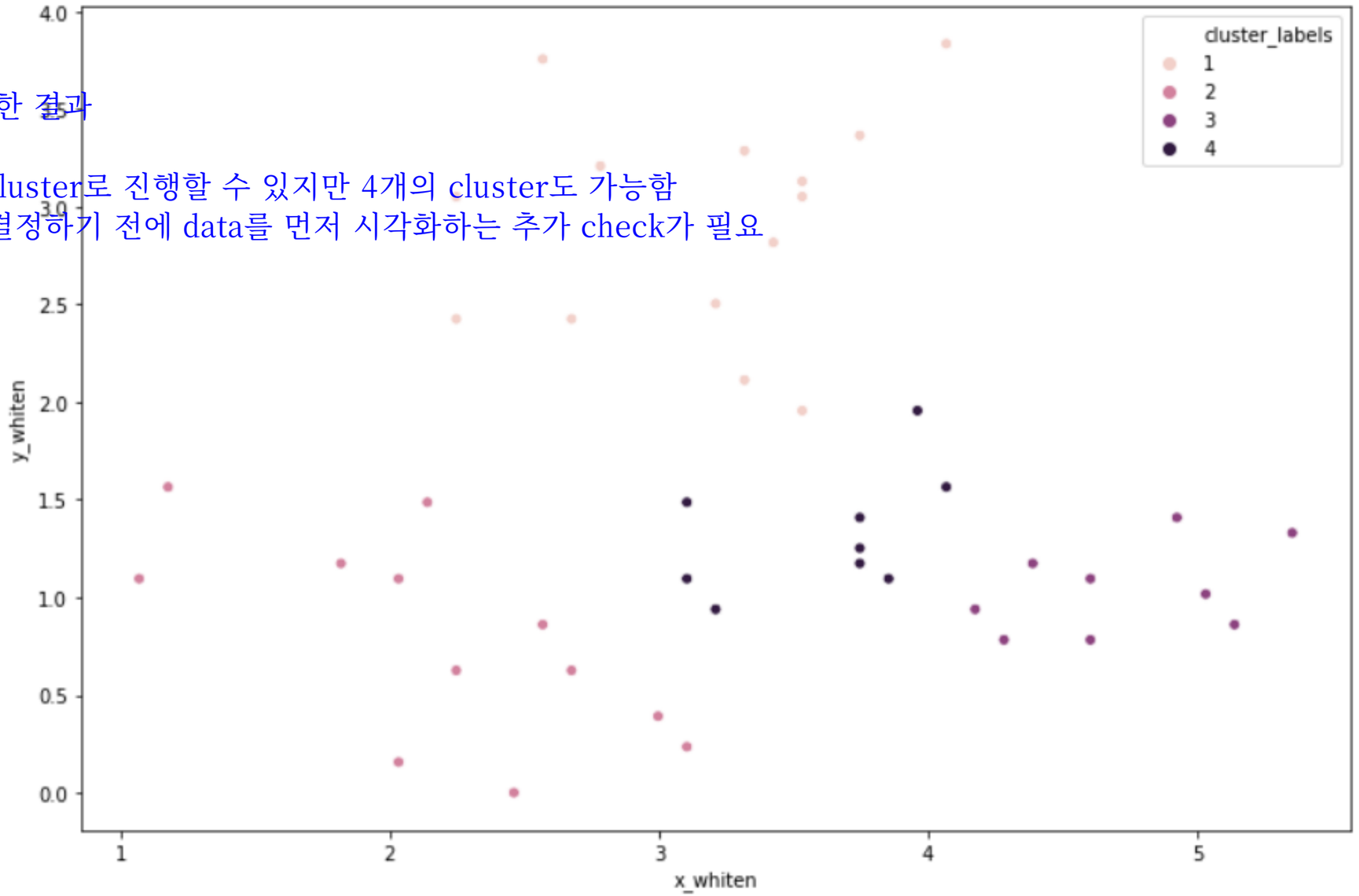


3개의 cluster를 사용한 결



4개의 cluster를 사용한 결과

덴더그램에서 3개의 cluster로 진행할 수 있지만 4개의 cluster도 가능함
따라서 cluster 수를 결정하기 전에 data를 먼저 시각화하는 추가 check가 필요



Next up - try some exercises

CLUSTER ANALYSIS IN PYTHON

Limitations of hierarchical clustering

CLUSTER ANALYSIS IN PYTHON

Shaumik Daityari
Business Analyst

datacamp

Measuring speed in hierarchical clustering

- `timeit` module
- Measure the speed of `.linkage()` method
- Use randomly generated points
- Run various iterations to extrapolate

반복에 걸리는 시간을 확인하기 위해 계층형 클러스터링의 다양한 반복 속도를 측정하는 작은 작업을 설계

- timeit module을 사용하여 함수 실행 시간을 확인.
- 계층 clustering process에서 가장 시간이 걸리는 단계인 linkage() method을 통해 거리 행렬을 구성하는 것이므로 행렬을 형성하는데 걸리는 시간을 측정
- 여기서 XY 평면에서 랜덤하게 생성된 데이터 포인트를 사용. 알고리즘의 한계를 테스트하기 위해 증가하는 데이터 포인트를 사용

Use of timeit module

다수의 점에 대해 시간이 얼마나 걸리는지 확인하기 위해 랜덤하게 생성된 좌표를 사용하여 100개의 점에 대해 링크 방법을 실행하는데 걸리는 시간을 살펴봄

```
from scipy.cluster.hierarchy import linkage
import pandas as pd
import random, timeit
points = 100
df = pd.DataFrame({'x': random.sample(range(0, points), points),
                   'y': random.sample(range(0, points), points)})
%timeit linkage(df[['x', 'y']], method = 'ward', metric = 'euclidean')
```

0~100 사이의 좌표 내에서 100개의 점 생성

interpreter에서 함수의 시간을 확인하려면 백분율을 사용.

timeit 모듈은 함수를 여러번 실행하고 런타임의 평균 및 표준 편차를 report

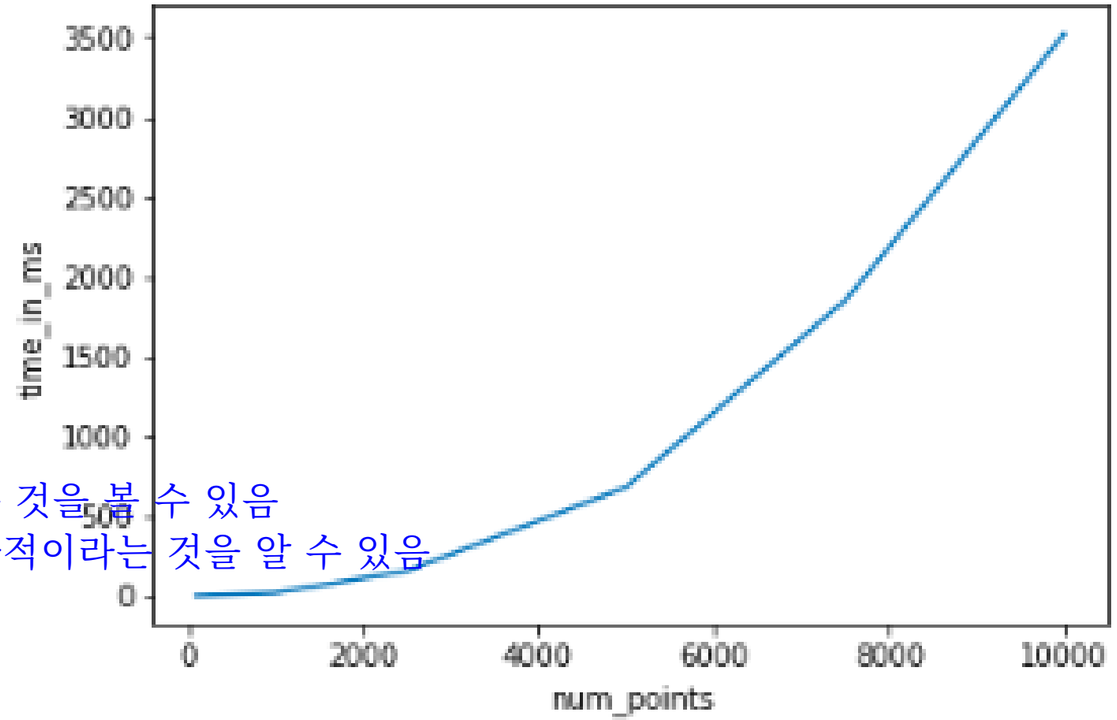
```
1.02 ms ± 133 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

jupyter notebook을 실행하는 경우 명령문을 실행하는 평균 시간은 약 1.02ms

증가하는 반복을 수행 = > point의 수, 연결 방법을 실행하는데 걸리는 시간을 확인하 다음 성능을 비교하기 위해 그래프를 plot

Comparison of runtime of linkage method

- Increasing runtime with data points
- Quadratic increase of runtime
- Not feasible for large datasets



- point의 수와 함께 data point의 수가 증가함에 따라 runtime 이 증가하는 것을 볼 수 있음
- 또한 런 시간의 증가는 데이터 점의 증가와 관련하여 선형적이지 않고 2차적이라는 것을 알 수 있음

이 때문에, 예를 들면, 월마트에 있는 모든 고객의 1년간 쇼핑 습관 등,
방대한 data point에 대해서 계층 clustering 기술을 실현할 수 없게 됨

Next up - exercises

CLUSTER ANALYSIS IN PYTHON