KU x depa
MACHINE LEARNING
Training Course 2022

Clustering Algorithms

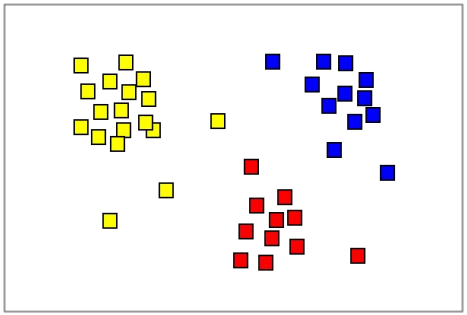
DR. SETHAVIDH GERTPHOL

KU KASETSART UNIVERSITY depa

Today's Outline

- Clustering algorithms
- K-Mean Clustering
- DBSCAN

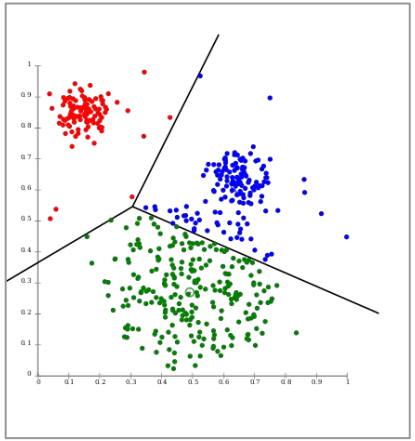
Unsupervised Machine Learning



Source: By Cluster-2.gif: hellispderivative work: Wgabrie (talk) - Cluster-2.gif, Public Domain, https://commons.wikimedia.org/w/index.php?curid=9442336

Clustering มีเป้าหมายเพื่อจะจัดกลุ่มตัวอย่างที่มีคุณลักษณะที่ใกล้เคียงกันให้อยู่รวมเป็นกลุ่มเดียวกันโดยคำนวณจากระยะห่างของแต่ละตัวอย่าง

- Medical patients
  - Feature values: อายุ, เพศ, อาการที่ 1, อาการที่ 2, ผลการรักษา 1, ผลการรักษา 2
- Web pages
  - Feature values: URL domain, length, #images, heading 1, heading 2, ..., heading n
- Products
  - Feature values: หมวดหมู่, ชื่อ, ขนาด, น้ำหนัก, ราคา



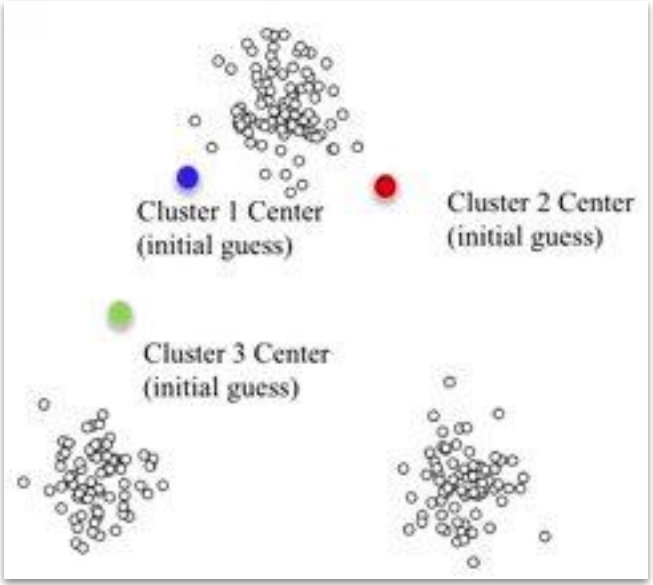
Source: By Chire - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=17085714

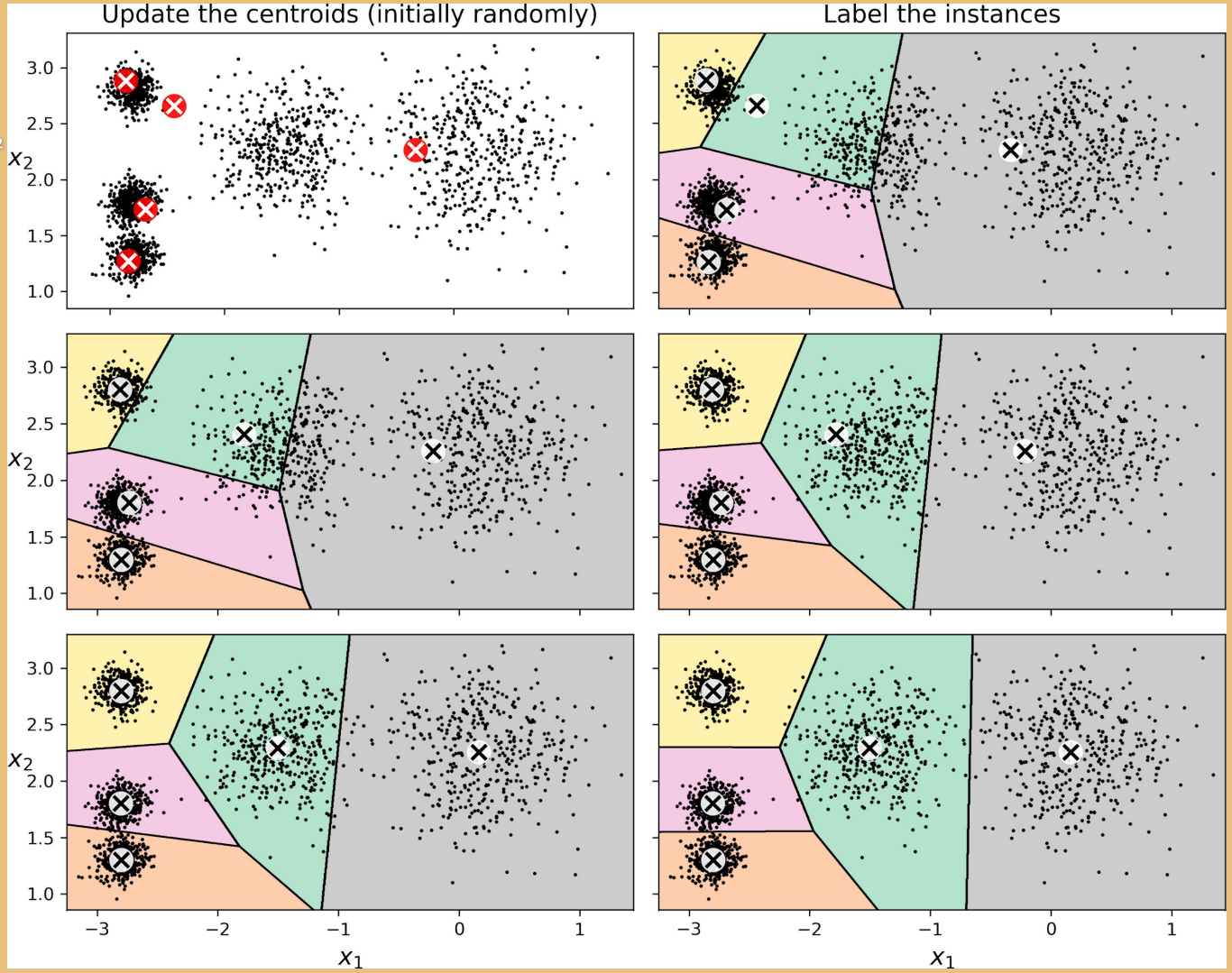
Clustering Algorithms

- แยกตัวอย่างออกเป็นกลุ่มๆ ตัวอย่างในแต่ละกลุ่มจะมีความ"คล้ายกัน" มากกว่ากับตัวอย่างนอกกลุ่ม
  - ไม่มี label ให้ โมเดลต้องจัดกลุ่มเอง (เป็น Unsupervised Learning)
- โมเดลจะให้รหัสกลุ่มกับทุกตัวอย่างเป็นผลลัพธ์ของโมเดล
- การจัดกลุ่มแบ่งเป็นสองประเภท
  - จัดกลุ่มแบบยาก: ข้อมูลแต่ละตัวอย่างจะอยู่ในกลุ่มเดียว
  - จัดกลุ่มแบบ fuzzy: ตัวอย่างอาจอยู่ในหลายกลุ่มได้ และจะมี weight หรือความน่าจะเป็นให้ว่าโอกาสที่ตัวอย่างนี้จะอยู่ในแต่ละกลุ่มเป็นเท่าใด

K-Means Clustering

- วิธีการจัดกลุ่มแบบง่าย
- กำหนดค่า K หมายถึง กำหนดจำนวนของกลุ่มที่ต้องการจะจัด และสุ่มตำแหน่งจุดกลางของแต่ละกลุ่ม
- คำนวนระยะทางระหว่างแต่ละตัวอย่างกับจุดกลางทั้งหมด
  และกำหนดให้ตัวอย่างนั้นอยู่กับกลุ่มที่ใกล้จุดกลางที่สุด
- คำนวนจุดกลางของแต่ละกลุ่มใหม่โดยใช้ค่าเฉลี่ยของสมาชิกกลุ่มทุกตัวอย่าง
- ทำซ้ำจนจุดกลางของทุกกลุ่มเข้าสถานะเสถียร




Update the centroids (initially randomly)
Label the instances
x_2
x_1
3.0
2.5
2.0
1.5
1.0
−3
−2
−1
0
1

In Sklearn

Import pandas เพื่อจัดการตาราง

Import seaborn เพื่อสร้างกราฟ

Pyplot ใช้ในการเพิ่มและปรับการแสดงกราฟ

สร้างตารางจากไฟล์ csv

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

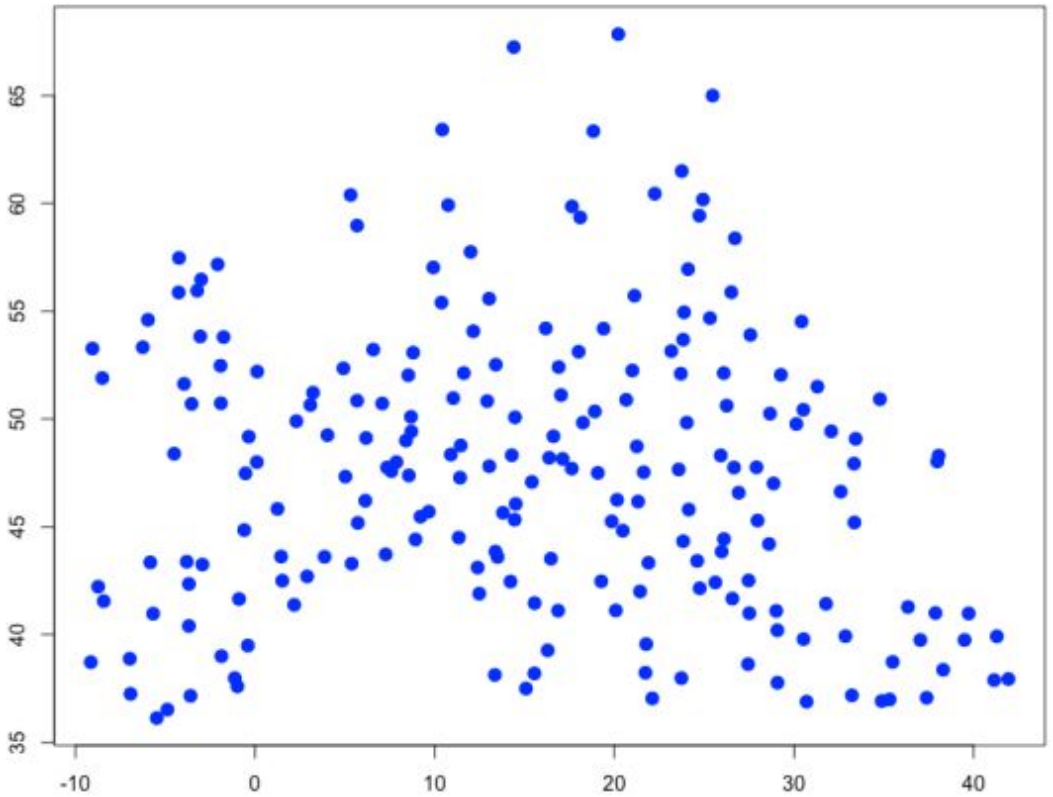
```
[3] df = pd.read_csv('/content/drive/MyDrive/Datasets/03-cities.csv', encoding='tis620')
    df.head()
```

	city	country	latitude	longitude	temperature	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Un
0	Aalborg	Denmark	57.03	9.92	7.52	NaN	NaN	NaN	NaN	
1	Aberdeen	United Kingdom	57.17	-2.08	8.10	NaN	NaN	NaN	NaN	
2	Abisko	Sweden	63.35	18.83	0.20	NaN	NaN	NaN	NaN	
3	Adana	Turkey	36.99	35.32	18.67	NaN	NaN	NaN	NaN	
4	Albacete	Spain	39.00	-1.87	12.62	NaN	NaN	NaN	NaN	

5 rows × 23 columns

K-Means Clustering

- Clustering European Cities



สร้างกราฟจากข้อมูล

```
sns.relplot(data=df, x='longitude', y='latitude', height=6, aspect=1.2)
```

KMeans

```
[5]  from sklearn.preprocessing import StandardScaler
```

```
[6]  scaler = StandardScaler()
     X_transformed = scaler.fit_transform(df[ ['latitude', 'longitude'] ])
```

Preprocessing ปรับค่าของคอลัมน์ให้อยู่ในช่วงเดียวกัน

```
[8]  from sklearn.cluster import KMeans
```

Import KMeans

```
[9]  km = KMeans(5)
     km.fit(X_transformed)
```

สร้าง KMeans โมเดลที่มีค่า K เท่ากับ 5

```
KMeans(n_clusters=5)
```

สั่ง fit เพื่อให้โมเดลเรียนรู้จากข้อมูล

ผลลัพธ์การจัดกลุ่ม

ผลลัพธ์อยู่ใน attribute ชื่อ labels_

```
[10] km.labels_

array([4, 3, 4, 2, 1, 1, 3, 3, 1, 1, 3, 2, 2, 0, 2, 3, 0, 1, 0, 0, 1, 2,
       3, 2, 3, 0, 1, 4, 3, 0, 3, 0, 1, 3, 3, 4, 1, 3, 1, 0, 3, 3, 1, 2,
       0, 3, 3, 0, 0, 3, 2, 0, 2, 1, 2, 0, 0, 3, 3, 1, 2, 3, 0, 0, 0, 0,
       2, 3, 2, 2, 0, 0, 2, 3, 3, 3, 3, 2, 2, 0, 3, 2, 2, 2, 3, 2, 3, 3,
       3, 2, 1, 1, 3, 4, 1, 0, 1, 3, 0, 3, 4, 0, 0, 1, 3, 3, 3, 2, 2, 2,
       3, 0, 2, 0, 0, 0, 4, 4, 0, 0, 0, 0, 3, 3, 1, 0, 1, 0, 0, 3, 1, 3,
       0, 2, 4, 2, 1, 1, 0, 2, 3, 1, 0, 1, 3, 1, 1, 2, 0, 3, 2, 0, 4, 0,
       4, 1, 1, 2, 1, 1, 1, 0, 2, 2, 2, 0, 0, 3, 4, 0, 0, 1, 3, 2, 1, 0,
       2, 1, 1, 0, 2, 2, 2, 2, 2, 4, 4, 0, 3, 0, 4, 4, 2, 4, 2, 1, 2, 0,
       2, 4, 4, 4, 1, 0, 1, 0, 0, 0, 2, 1, 0, 2, 3], dtype=int32)
```

ผลลัพธ์คือเบอร์กลุ่มของแต่ละตัวอย่าง

เบอร์ของกลุ่มไม่มีความหมาย และอาจเปลี่ยนไปเมื่อใช้คำสั่ง fit อีกครั้ง

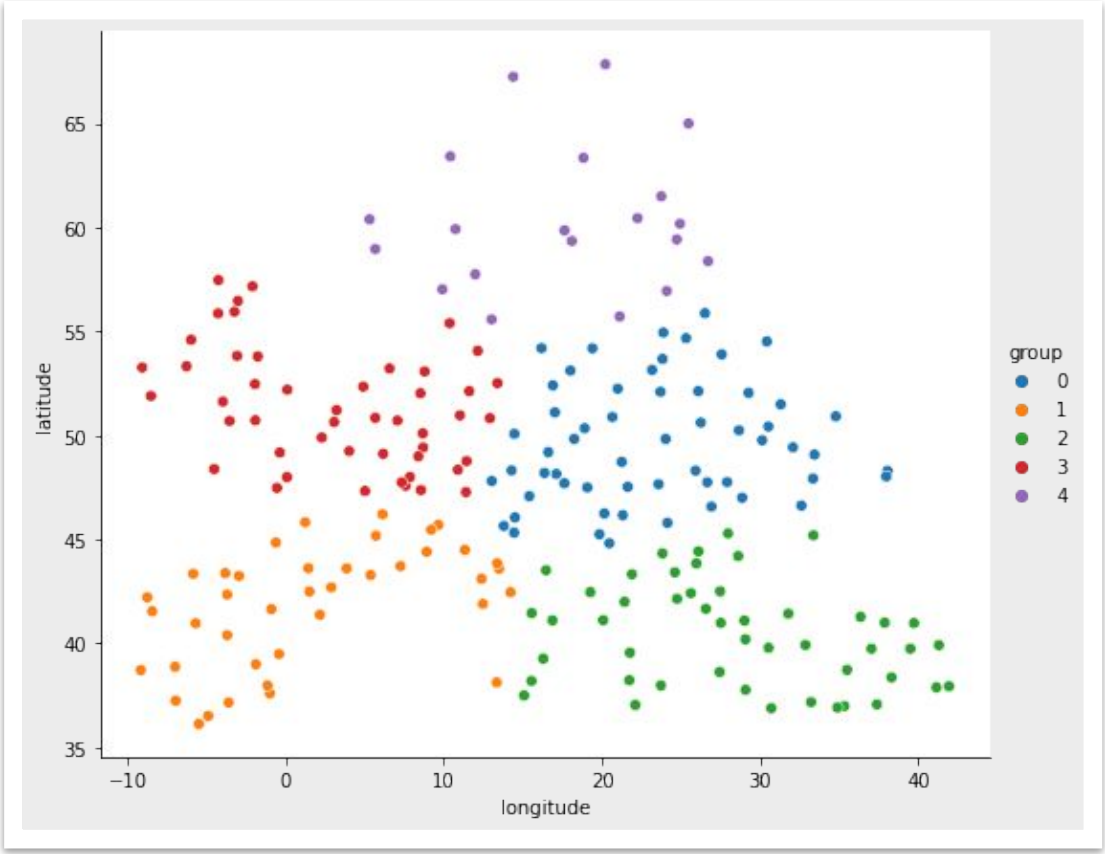
นำเบอร์กลุ่มไปสร้างเป็นคอลัมน์ในตาราง

```
[31] city_df = df.iloc[:, :5].join(pd.Series(km.labels_, name='group', dtype='category'))
     city_df.head()
```

	city	country	latitude	longitude	temperature	group
0	Aalborg	Denmark	57.03	9.92	7.52	4
1	Aberdeen	United Kingdom	57.17	-2.08	8.10	3
2	Abisko	Sweden	63.35	18.83	0.20	4
3	Adana	Turkey	36.99	35.32	18.67	2
4	Albacete	Spain	39.00	-1.87	12.62	1

K-Means Clustering

- K = 5




```
[33] sns.relplot(data=city_df, x='longitude', y='latitude', hue='group',
                 height=6, aspect=1.2)
```

ศูนย์กลางกลุ่ม

```
[34] km.cluster_centers_
```

```
array([[ 0.3169463 ,  0.61676444],
       [-0.84185202, -1.04395218],
       [-1.04625325,  0.99412297],
       [ 0.56996343, -0.92349972],
       [ 1.92350263,  0.16062993]])
```

ศูนย์กลางกลุ่มอยู่ใน attribute ชื่อ cluster_centers_

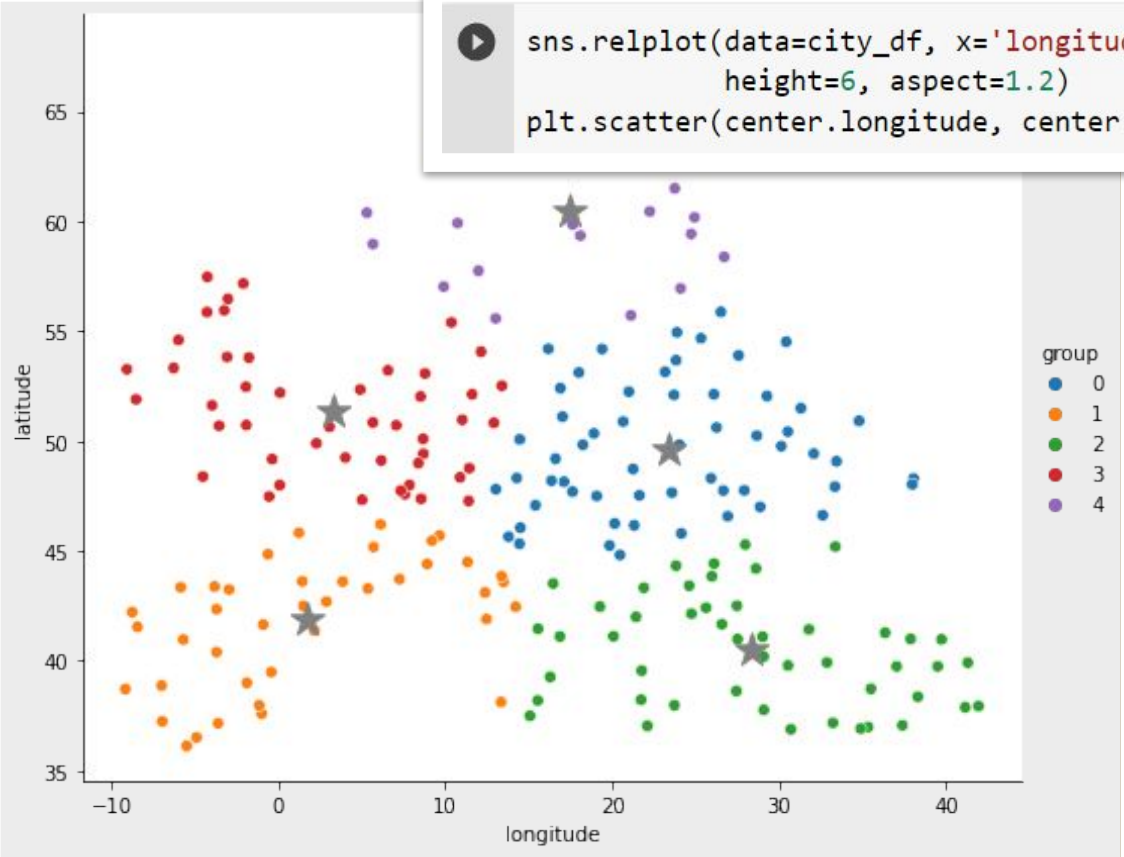
```
center = pd.DataFrame(scaler.inverse_transform(km.cluster_centers_),
                      columns=['latitude', 'longitude']
)
center
```

	latitude	longitude
0	49.628246	23.415263
1	41.846750	1.764000
2	40.474167	28.335000
3	51.327292	3.334375
4	60.416500	17.468500

Inverse transform ให้ค่ากลับมาเป็น latitude กับ longitude

K-Means Clustering

```
sns.relplot(data=city_df, x='longitude', y='latitude', hue='group',
            height=6, aspect=1.2)
plt.scatter(center.longitude, center.latitude, marker='*', color='gray', s=300)
```



Performance evaluation

- การจัดกลุ่มด้วย KMeans แต่ละครั้งอาจได้ผลไม่เหมือนกัน ขึ้นกับการสุ่มศูนย์กลางกลุ่มตอนเริ่มต้น
- จัดกลุ่มครั้งไหนดีกว่ากัน?
- Inertia: ผลบวกของระยะห่างยกกำลังสองระหว่างตัวอย่างและศูนย์กลางกลุ่ม (ยิ่งต่ำยิ่งดี)
- ใช้ option n_init เพื่อกำหนดจำนวนครั้งที่จะลองสร้าง KMeans แล้วเก็บผลของครั้งที่ inertia ต่ำที่สุด
- Inertia ไม่สามารถใช้เลือกจำนวน K ที่เหมาะที่สุดได้โดยตรง เพราะค่า inertia มักจะลดลงเมื่อเราเพิ่ม K

```
[67] km.inertia_

     95.6254876810319
```

```
km2 = KMeans(5)
km2.fit(X_transformed)
km2.inertia_

95.49171488121473
```

```
[13] km_best_inertia = KMeans(5, n_init=100)
     km_best_inertia.fit(X_transformed)
     km_best_inertia.inertia_

     95.4768479499298
```

```
[70] km3 = KMeans(8)
     km3.fit(X_transformed)
     km3.inertia_

     52.10909936218606
```

Finding best value of K

- Plot K vs. inertia and look for an elbow

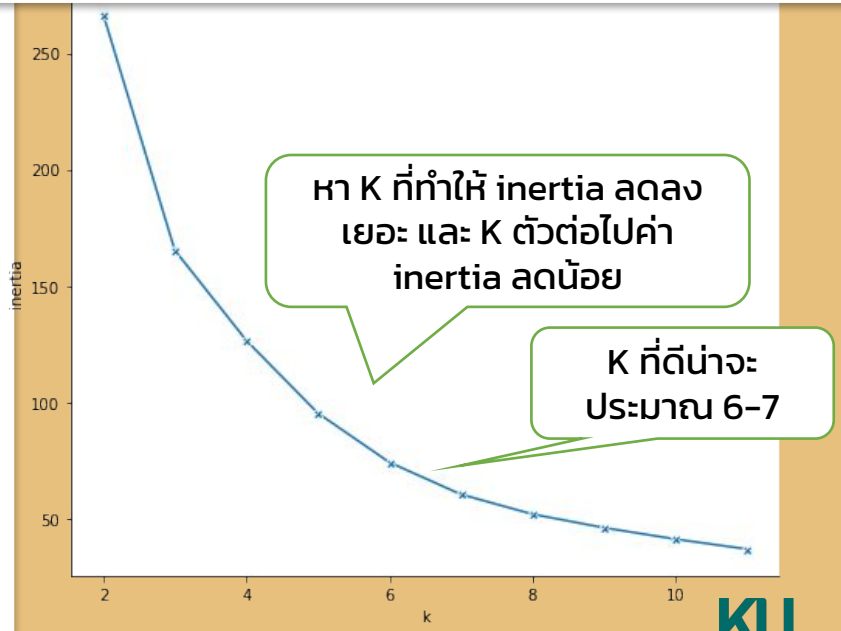
```
[14] results = []
     for k in range(2,12):
       km_search = KMeans(k, n_init=100)
       km_search.fit(X_transformed)
       results.append( (k, km_search.inertia_))
```

```
result_df = pd.DataFrame(results, columns=['k','inertia'])
result_df
```

	k	inertia
0	2	266.544171
1	3	165.375146
2	4	126.656603
3	5	95.464827
4	6	74.329201
5	7	60.590514
6	8	52.078978

ปรับค่า K จาก 2-12 เก็บ inertia ที่ดีที่สุด

```
sns.relplot(data=result_df, x='k', y='inertia', kind='line',
            height=6, aspect=1.2, marker='X')
```



หา K ที่ทำให้ inertia ลดลงเยอะ และ K ตัวต่อไปค่า inertia ลดน้อย

K ที่ดีน่าจะประมาณ 6-7

Silhouette coefficient

- ค่า silhouette ของตัวอย่างคำนวณจาก (b-a)/max(a,b) โดย
  - a: ระยะห่างเฉลี่ยของตัวอย่างนั้นไปยังตัวอย่างอื่นในคลัสเตอร์เดียวกัน
  - b: ระยะห่างเฉลี่ยของตัวอย่างนั้นไปยังตัวอย่างทุกตัวในคลัสเตอร์อื่นที่ใกล้ที่สุด
- ยิ่งค่า silhuoette ที่ใกล้ +1 หมายถึงตัวอย่างนี้อยู่ในกลุ่มที่เหมาะสมแล้ว
- ค่า silhuoette ที่ใกล้ 0 หมายถึงตัวอย่างนี้อยู่ตรงขอบของกลุ่ม
- ค่า silhuoette ที่ใกล้ -1 หมายถึงตัวอย่างนี้น่าจะอยู่กลุ่มอื่น

In Sklearn

import

```
[17] from sklearn.metrics import silhouette_score
```

เปลี่ยน K จาก 2-12 แล้วเก็บค่า silhouette

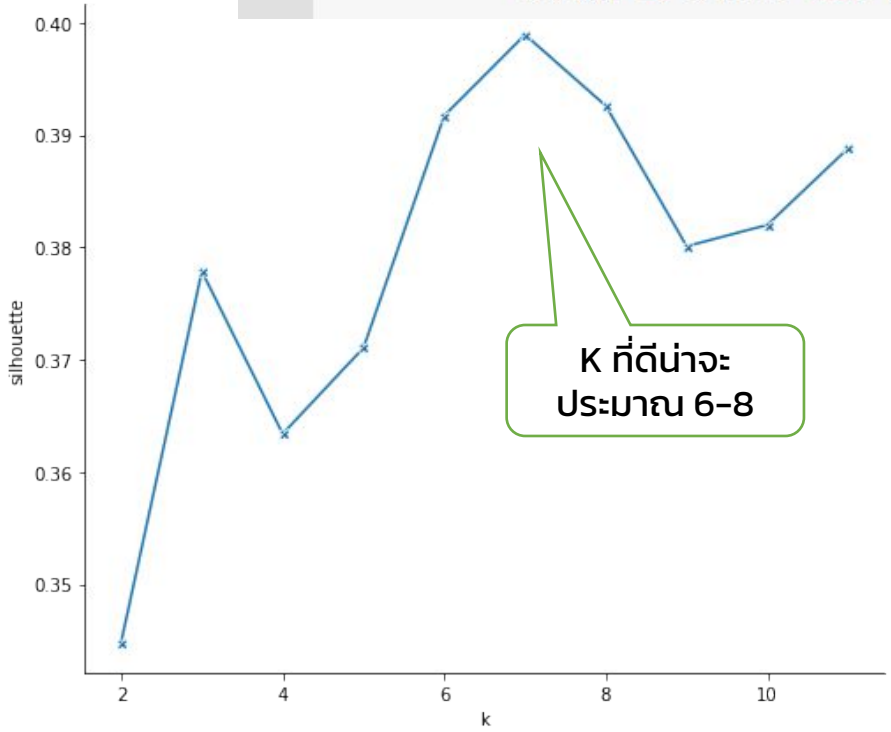
```
[18] results = []
     for k in range(2,12):
       km_search = KMeans(k, n_init=100)
       km_search.fit(X_transformed)
       results.append( (k, silhouette_score(X_transformed, km_search.labels_)))
```

```
silhouette_df = pd.DataFrame(results, columns=['k','silhouette'])
silhouette_df
```

	k	silhouette
0	2	0.344822
1	3	0.377867
2	4	0.363395
3	5	0.371055

Plot กราฟหา K ที่เหมาะสม

```
sns.relplot(data=silhouette_df, x='k', y='silhouette', kind='line',
            height=6, aspect=1.2, marker='X')
```



ข้อดีข้อเสียของ K-Mean Clustering

ข้อดี

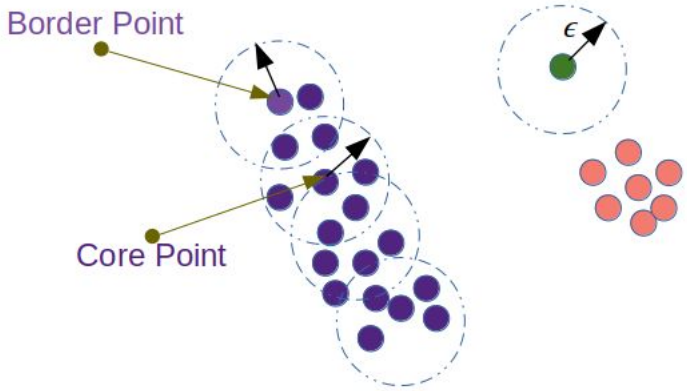
- เข้าใจง่าย ใช้ง่าย
- เหมาะกับกลุ่มที่มีจำนวนข้อมูล ใกล้เคียงกัน และมีรูปร่างเป็น ทรงกลม

ข้อเสีย

- เซ็นซิทิฟต่อตำแหน่งเริ่มต้นของจุดกลางของกลุ่ม
- ต้องปรับค่า feature ให้อยู่ในสเกลเดียวกัน
- ใช้ไม่ได้กับ feature ที่เป็น category
- ไม่เหมาะกับกลุ่มที่มีลักษณะเป็นวงรียาว หรือจำนวนข้อมูลแตกต่างกันมาก ในแต่ละกลุ่ม

DBSCAN

- เทคนิคในการแบ่งกลุ่มโดยดูตามความหนาแน่นของข้อมูล
- กำหนดพื้นที่ในการวิเคราะห์เป็นวงกลมรัศมี epsilon (ε) และจำนวนข้อมูลที่น้อที่สุดในการนับเป็นพื้นที่หนาแน่น
- จุดที่อยู่ในพื้นที่ที่ข้อมูลหนาแน่นจะเรียก Core
- จุดที่ติดกับพื้นที่หนาแน่นแต่จำนวนข้อมู ไม่พอจะเรียกว่า Border Point
- จุดที่ไม่ติดกับพื้นที่หนาแน่นเรียกว่า Noise



$$N_{Eps}(p) = \{q \in D \text{ such that } dist(p,q) \leq \epsilon\}$$

ϵ = 1 unit, MinPts = 7

https://towardsdatascience.com/dbscan-algorithm-complete-guide-and-application-with-python-scikit-learn-d690cbae4c5d

In Sklearn

```
[27] from sklearn.cluster import DBSCAN
```

import

```
[42] dbs = DBSCAN(eps=.27, min_samples=5)
     dbs.fit(X_transformed)
     dbs.labels_
```

eps เป็นค่ารัศมี (ε)

min_sample คือจำนวนตัวอย่างที่น้อยที่สุดที่จะนับเป็นกลุ่ม

เบอร์กลุ่มอยู่ใน labels_

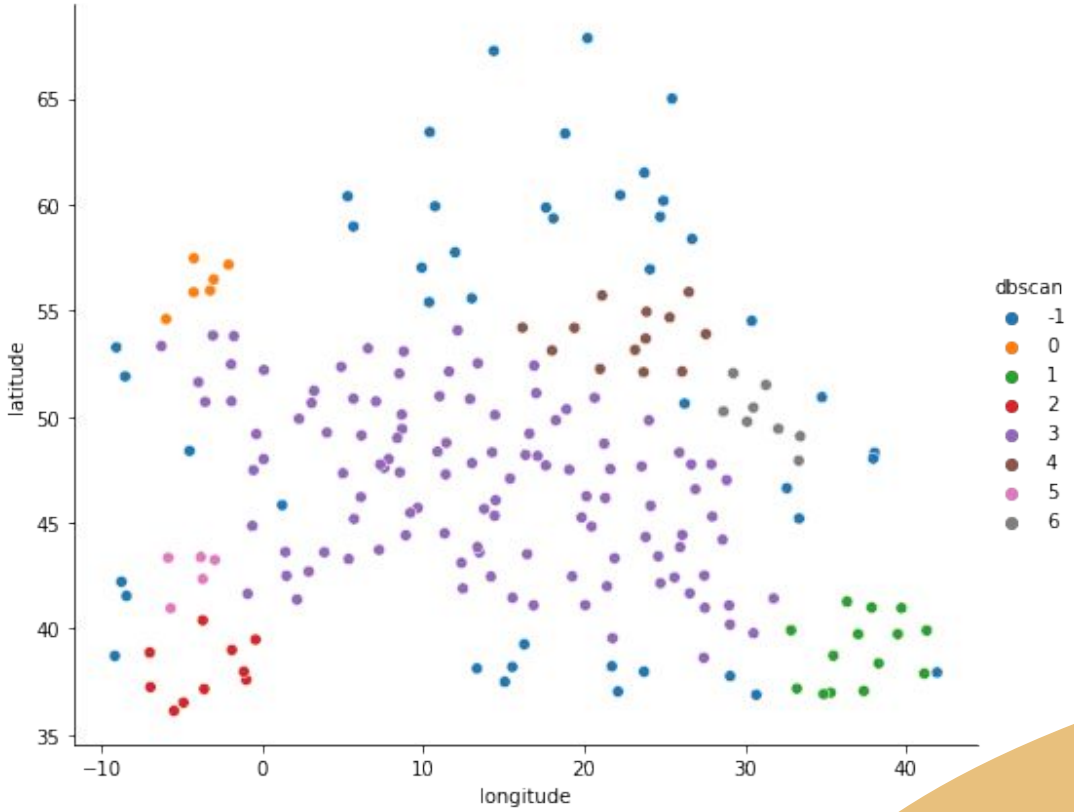
```
array([-1,  0, -1,  1,  2,  2,  3,  3,  3,  3,  3,  1, -1,  3, -1,  3,  3,
        2,  3,  3,  3,  3,  3,  1,  0,  3,  3, -1,  3,  4,  3,  6,  5,  3,
        3, -1,  3,  3,  3,  3,  3,  3, -1,  3,  3,  3, -1,  4,  3,  3,  3,
        3,  3,  5,  3,  4,  3,  3,  3,  2, -1,  3,  6,  6,  3,  3,  3, -1,
       -1,  3,  4,  3, -1,  3,  3,  0,  0,  3,  3,  4,  3,  1,  1,  3,  3,
        3,  3,  3, -1,  1,  3,  3,  0, -1,  2,  3,  3,  3,  3,  3, -1, -1,
        4,  2,  3,  3,  0,  3, -1,  1,  3,  4,  1, -1,  3,  6, -1,  4,  3,
        4,  6,  6,  3,  3, -1,  3, -1,  3,  3,  3,  2,  3, -1,  1, -1,  3,
        2,  3,  6, -1,  3,  3,  4,  3,  3,  2,  3,  3,  3, -1,  1, -1, -1,
        3, -1,  5, -1, -1,  3,  3,  3,  4,  3,  3,  3,  3,  3,  3, -1,  3,
       -1,  3,  3,  3,  5,  3,  1,  5,  3,  3, -1,  1,  3,  3,  3, -1, -1,
       -1,  3,  3, -1, -1,  1, -1,  3,  3,  1,  3,  3, -1, -1, -1,  2,  3,
       -1,  4,  4,  3, -1,  3,  6,  3,  3])
```

```
city_df = city_df.join(pd.Series(dbs.labels_, name='dbscan', dtype='category'))
city_df
```

	city	country	latitude	longitude	temperature	group	dbscan
0	Aalborg	Denmark	57.03	9.92	7.52	3	-1
1	Aberdeen	United Kingdom	57.17	-2.08	8.10	4	0

DBSCAN Example

- ε = 0.27
- จำนวนจุด = 5




```
sns.relplot(data=city_df, x='longitude', y='latitude', hue='dbscan',
            height=6, aspect=1.2)
```

Measuring performance

- ใช้ silhouette score ได้ แต่ไม่ควรใช้ inertia เพราะค่านี้มี bias กับกลุ่มที่เป็นทรงกลม

- ดูคะแนน silhouette ของแต่ละตัวอย่างได้โดยใช้ silhouette_samples

คะแนนใกล้ 0 เพราะกลุ่ม -1 (ไม่มีกลุ่ม) คร่อมกลุ่มอื่น ๆ

```
[52] silhouette_score(X_transformed, dbs.labels_)

     -0.029979953795536547
```

```
[57] from sklearn.metrics import silhouette_samples

[60] silhouette_samples(X_transformed, dbs.labels_)

     array([-0.4201391 ,  0.87472275, -0.32968403,  0.78945119,  0.45947117,
             0.59806587, -0.11131015, -0.28822401,  0.30721408, -0.65488582,
            -0.39855263,  0.68909483, -0.77375643, -0.01764048, -0.56710836,
             0.29166124, -0.48484691,  0.32997335, -0.35116554, -0.64737017,
            -0.61052259,  0.26621481,  0.21043136,  0.76840881,  0.79685883,
             0.13852788,  0.23065672, -0.56861582, -0.33803142,  0.58659648,
            -0.00386542,  0.724995  ,  0.73745388, -0.62055527, -0.77742282,
            -0.20953601,  0.25565654,  0.09710473, -0.6712225 , -0.57967451,
            -0.42158724, -0.74759587, -0.86889657, -0.41027484,  0.08677381,
            -0.12694005, -0.63810368,  0.39608723, -0.06536919, -0.24140712,
            -0.23051186,  0.08880183, -0.31589003,  0.70877981, -0.57676855,
```

ข้อดีข้อเสียของ DBSCAN

- ข้อดี
  - จัดกลุ่มตามความหนาแน่นของชุดข้อมูล
  - สามารถจัดกลุ่มที่ขนาดของกลุ่มไม่เท่ากันได้ดี
  - หาข้อมูลที่ไม่เข้ากลุ่มได้
  - ไม่ต้องระบุจำนวนกลุ่มในตอนแรก
- ข้อเสีย
  - การจัดกลุ่มขึ้นกับค่า ε กับจำนวนข้อมูลน้อยสุดในกลุ่ม อาจต้องปรับจูนค่าหลายรอบ
  - หากลุ่มที่มีความหนาแน่นต่างกันไม่ได้

References

- Aurélien Géron, "Hands-On Machine Learning with Scikit-Learn and TensorFlow", O'Reilly Media, Inc., March 2017.