1. **Pendahuluan Clustering**
2. **Video EDA-03**
3. **Code EDA-03**
4. **Referensi**

[OC] Machine learning

"We must be careful not to confuse data with the abstractions we use to analyse them" - *William James*

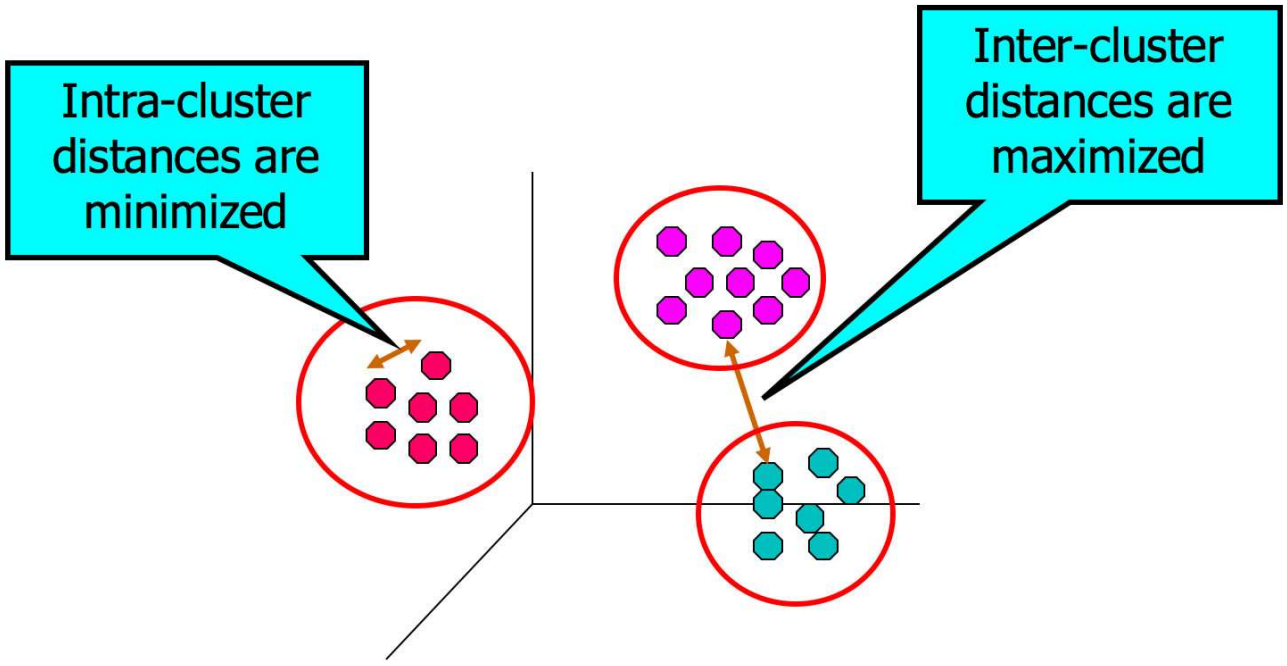
```
import warnings; warnings.simplefilter('ignore')
import umap, numpy as np, tau_unsup as tau, matplotlib.pyplot as plt, pandas as pd,
seaborn as sns
from sklearn import cluster, datasets
from sklearn.metrics import silhouette_score as siluet
from sklearn.metrics.cluster import homogeneity_score as purity
from sklearn.metrics import normalized_mutual_info_score as NMI

sns.set(style="ticks", color_codes=True)
random_state = 99
```

> An intelligent being cannot treat every object it sees as a unique entity unlike anything else in the universe. It has to put objects in categories so that it may apply its hard-won knowledge about similar objects encountered in the past, to the object at hand.
>
> Steven Pinker, *How the Mind Works*, 1997.

Definition

Clustering is as a process of finding group structures within data such that each instance within a group is similar to one another and dissimilar to instances in other groups [1]



[1]. Jain, A.K., Data clustering: 50 years beyond K-means. Pattern Recognition Letters, 2010. 31(8): p. 651-666.

Applications

Clustering analysis applications can be divided into two broad categories:

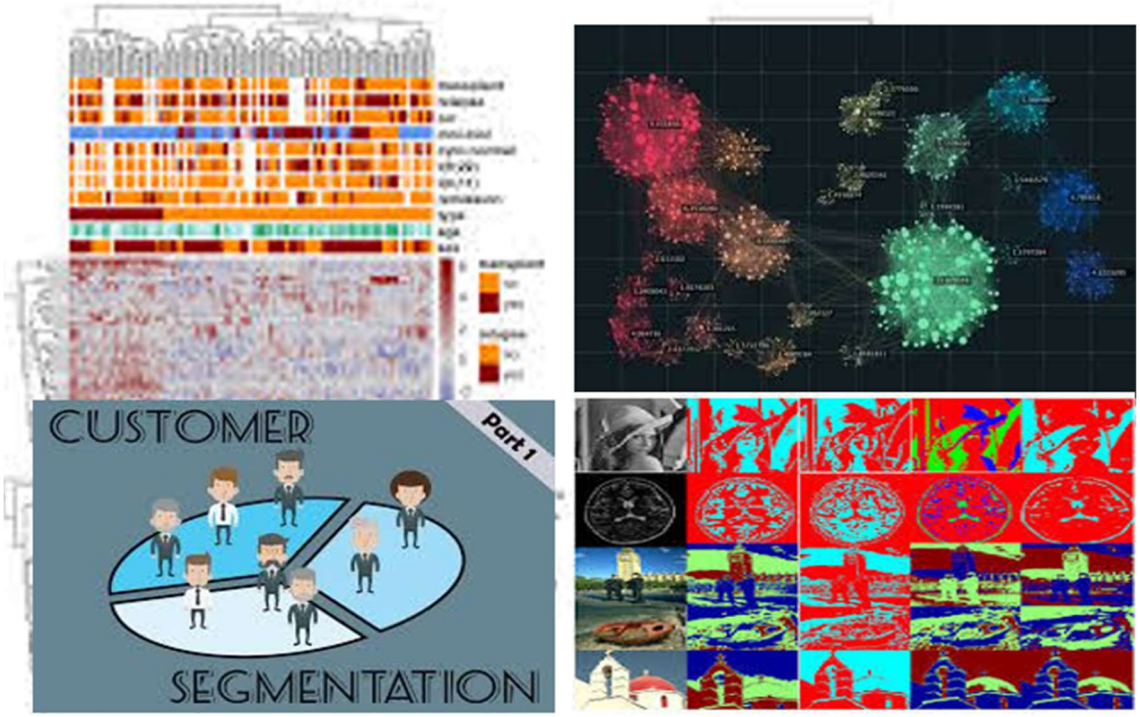
- clustering for utility (e.g., data compression and indexing) and
- clustering for understanding data (e.g., finding latent structures or insights in the data)

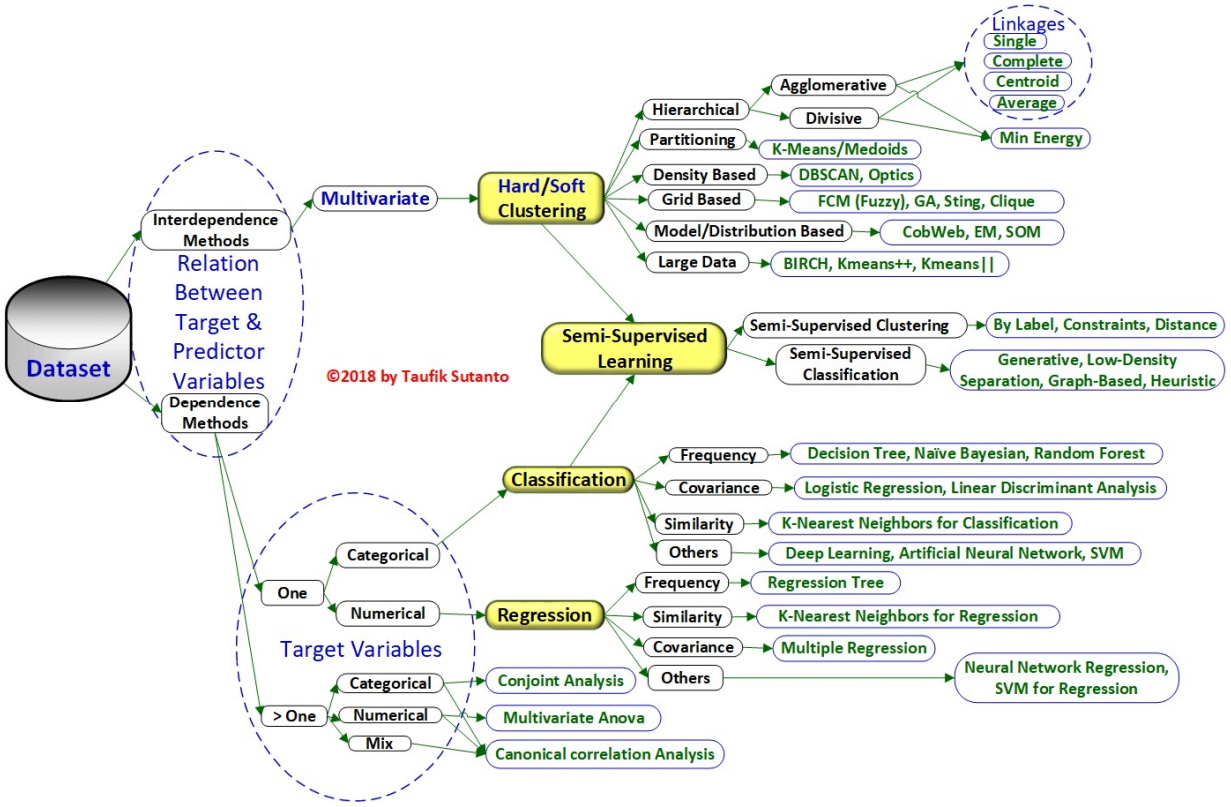
Methods developed in Data Mining fall into the second category.

[2]. Pang-Ning, T., M. Steinbach, and V. Kumar, Introduction to data mining. Vol. 74. 2006, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Realworld Clustering Applications

- Recommendation engines
- Market segmentation
- Social network analysis
- Search result grouping
- Medical imaging
- Image segmentation
- Anomaly detection





k-Means

Source: https://imgflip.com/

Algoritma k-Means

1. Initialize **cluster centroids** $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.
2. Repeat until convergence: {

   For every i, set
   $$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$$

   For each j, set
   $$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}.$$

   }

- How it works: https://www.learndatasci.com/tutorials/k-means-clustering-algorithms-python-intro/

Penting:

- Apakah pengaruh menggunakan centroid dan algoritma ini terhadap bentuk cluster?
- Dari pertanyaan sebelumnya pahami bias memilih algoritma clustering.
- k-Means tidak Robust terhadap outlier, Mengapa?
- Lalu apa yang sebaiknya dilakukan?<

Tantangan Clustering

- Computational Complexity
- Evaluation
- Interpretation
- Heavily depends on domain knowledge

In [3]:

```
# Kita akan menggunakan 2 data: [1]. Iris dan [2]. Data untuk Studi Kasus (segmentasi
kustomer) - di bagian akhir
# load the iris data

df = sns.load_dataset("iris")
X = df[['sepal_length','sepal_width','petal_length','petal_width']].values
C = df['species'].values
print(X.shape)
df.head()
```

```
(150, 4)
```

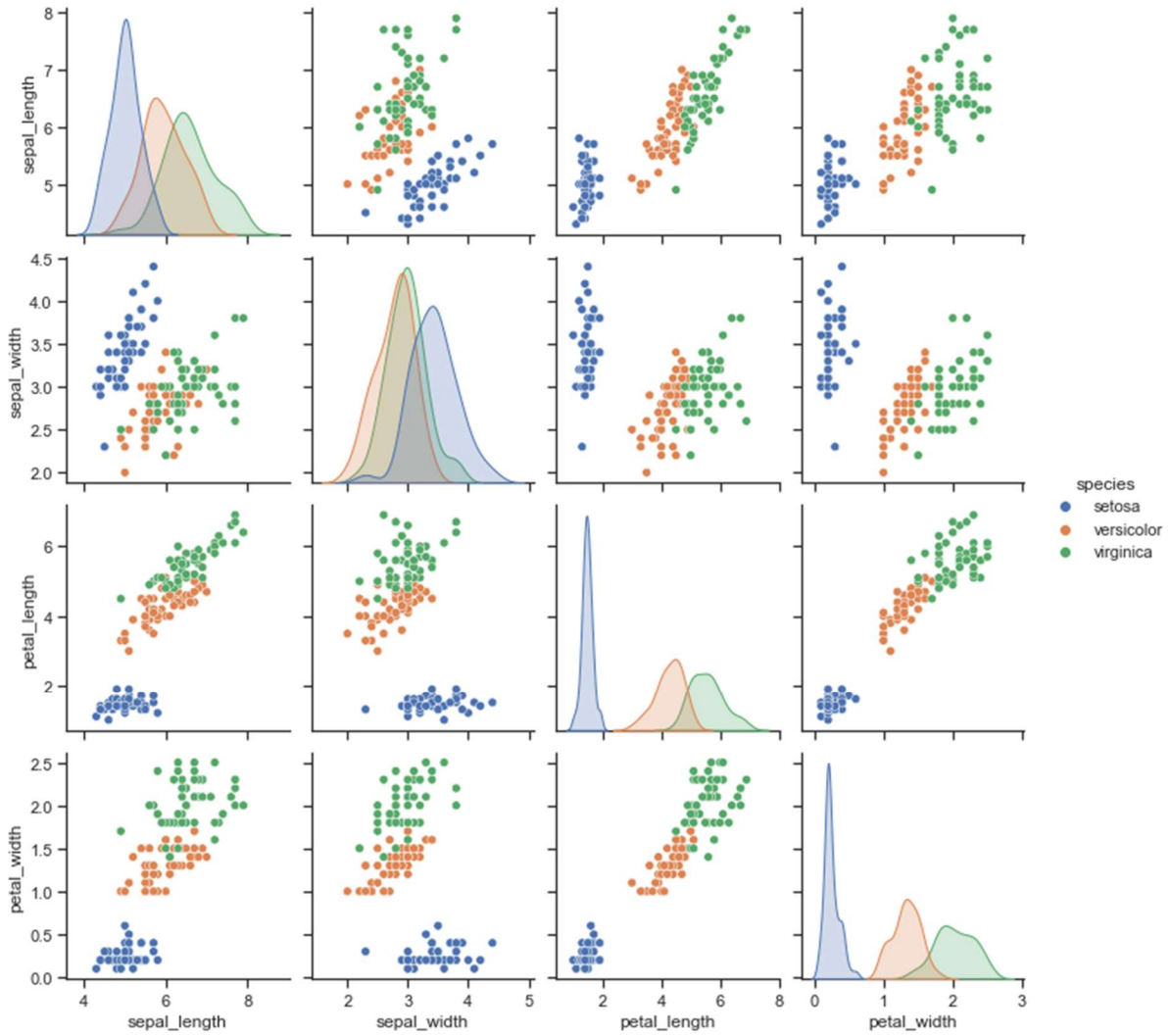
Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa

	sepal_length	sepal_width	petal_length	petal_width	species
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [4]:

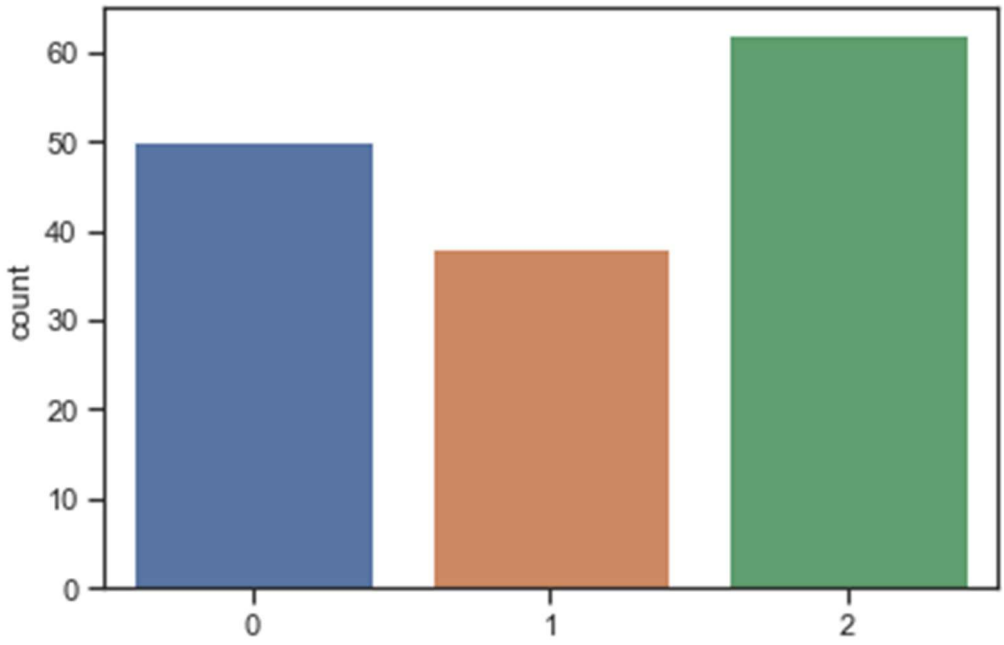
```
g = sns.pairplot(df, hue="species")
```



In [5]:

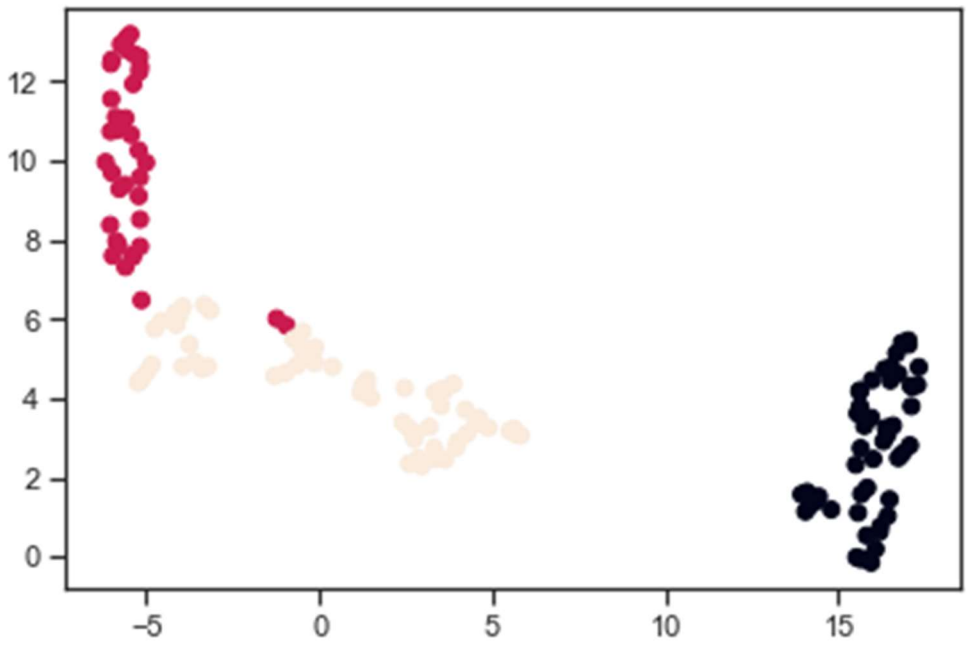
```
# k-means: http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans
# Hapus "random_state = random_state" jika ingin melihat efek randomized centroid.
k = 3
km = cluster.KMeans(n_clusters=k, init='random', max_iter=300, tol=0.0001, n_jobs=-1, random_state = random_state)
km.fit(X)
# Hasil clusteringnya
```

```
C_km = km.predict(X)
p= sns.countplot(C_km)
```



In [6]:

```
X2D = umap.UMAP(n_neighbors=5, min_dist=0.3, random_state=random_state).fit_transform(X)
fig, ax = plt.subplots()
ax.scatter(X2D[:,0], X2D[:,1], c=C_km)
plt.show()
```



Apa beda label ini dengan klasifikasi ("labels")?

In [7]:

```
C_km
```

Out[7]:

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

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1,
1, 1, 1, 2, 2, 1, 1, 1, 1, 2, 1, 2, 1, 2, 1, 1, 2, 2, 1, 1, 1, 1,
1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 2])
```

Evaluasi? - Inertia : Intra Cluster Distance

- Bagaimana memaknainya?
- Bukan Error! ... Mengapa?
- Belum ada faktor "inter distance" ==> nanti Silhouette Score

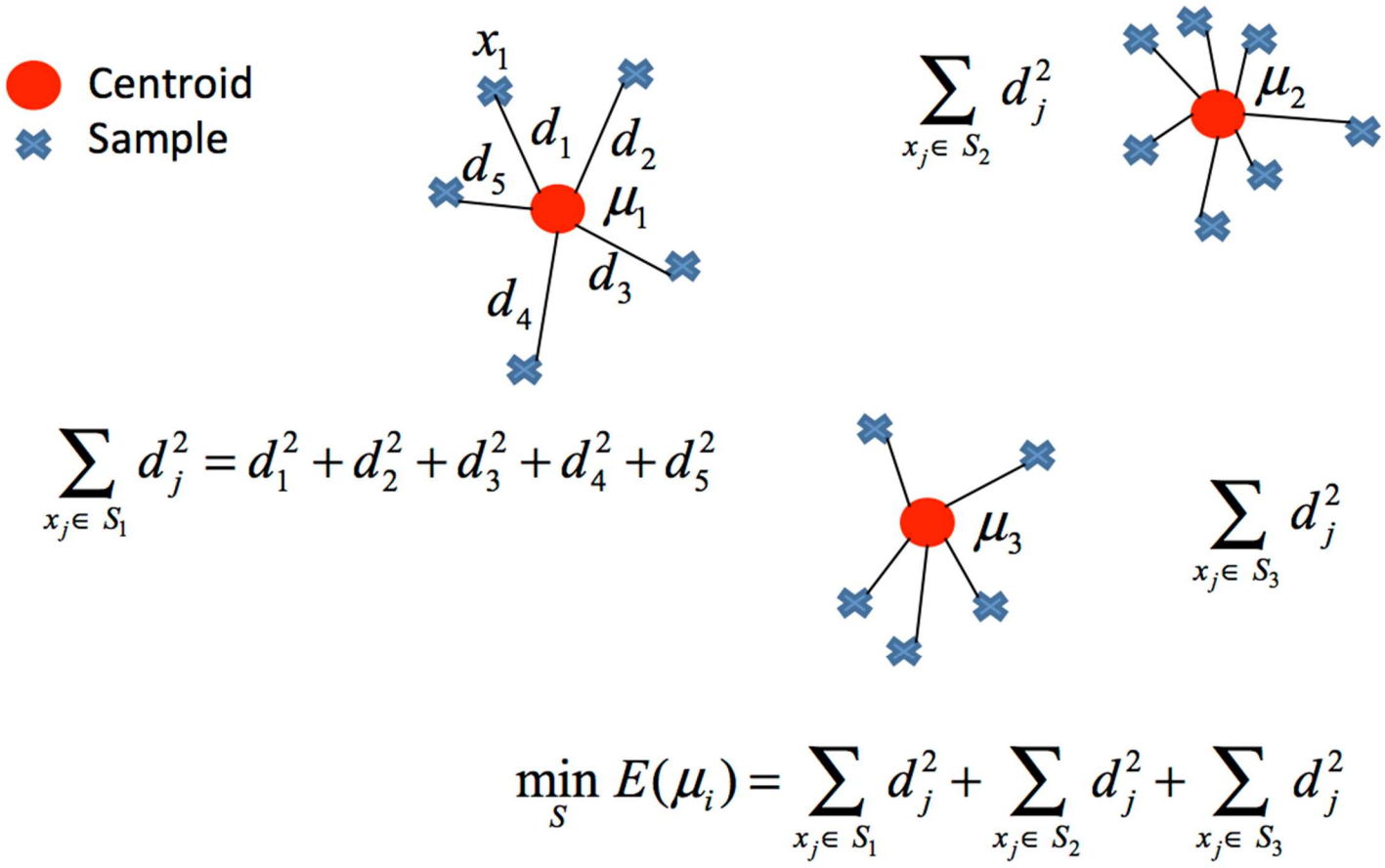


image source: https://www.unioviedo.es/compnum/labs/new/kmeans.html

In [8]:

```
km.inertia_
```

Out[8]:

```
78.851441426146
```

Optimal Number of Clusters? - Elbow Method -

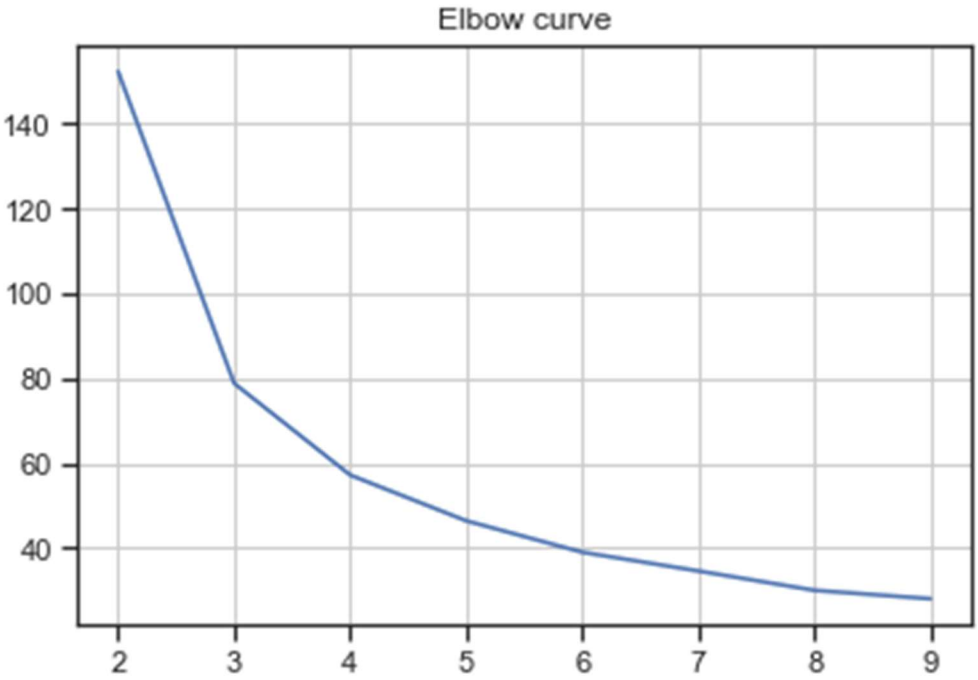
- Menggunakan inertia
- Rekomendasi ... Bukan "wajib" ==> Lalu apa yang lebih penting?

In [9]:

```
distorsions, k1, kN = [], 2, 10
for k in range(k1, kN):
    kmeans = cluster.KMeans(n_clusters=k).fit(X)
    distorsions.append(kmeans.inertia_)
#fig = plt.figure(figsize=(15, 5))
plt.plot(range(k1, kN), distorsions); plt.grid(True)
plt.title('Elbow curve')
```

Out[9]:

```
Text(0.5, 1.0, 'Elbow curve')
```



Ponder this

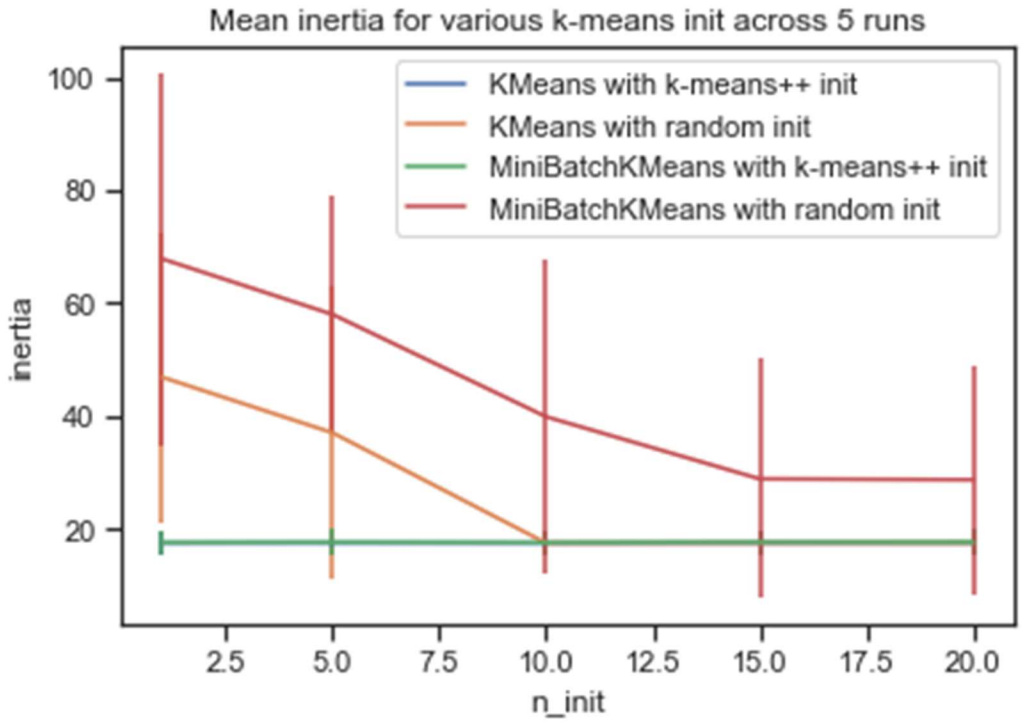
Apakah akibat dari mengacak (randomized) centroid di awal algoritma?

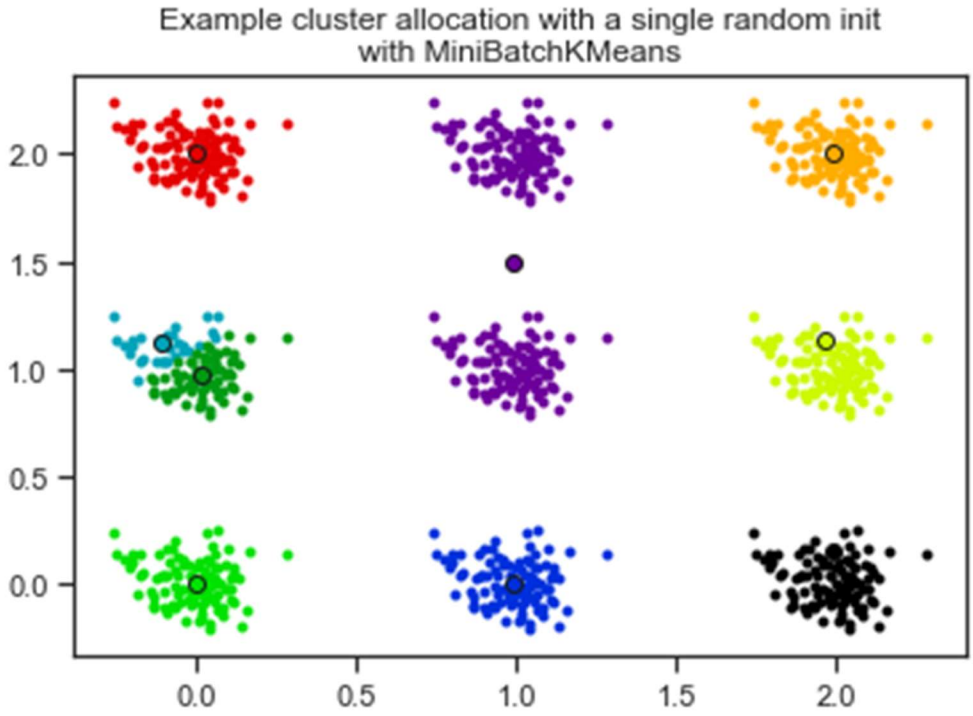
k-Means sangat tidak direkomendasikan untuk diaplikasikan di aplikasi nyata ... Loh? Mengapa?

In [10]:

```
tau.km_initializations()
```

```
Evaluation of KMeans with k-means++ init
Evaluation of KMeans with random init
Evaluation of MiniBatchKMeans with k-means++ init
Evaluation of MiniBatchKMeans with random init
```





k-Means++

- Original *k-means* memulai algoritmanya dengan mengacak centroid awal dan k-means tidak "robust" terhadap centroid awal ini (apa artinya?).
- **k-Means akan menghasilkan hasil yang berbeda-beda jika di-run beberapa kali!....**
- k-Means++ "mengatasi" hal ini:
- inisialisasi centroid tidak random, tapi dengan menghitung probabilitas terbaik bagi centroid awal.
- Keuntungan selain lebih robust, biasanya iterasi yang dibutuhkan jauh lebih sedikit ketimbang k-means biasa.
- Reference : http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf

image Source: https://medium.com/@phil.busko/animation-of-k-means-clustering-31a484c30ba5

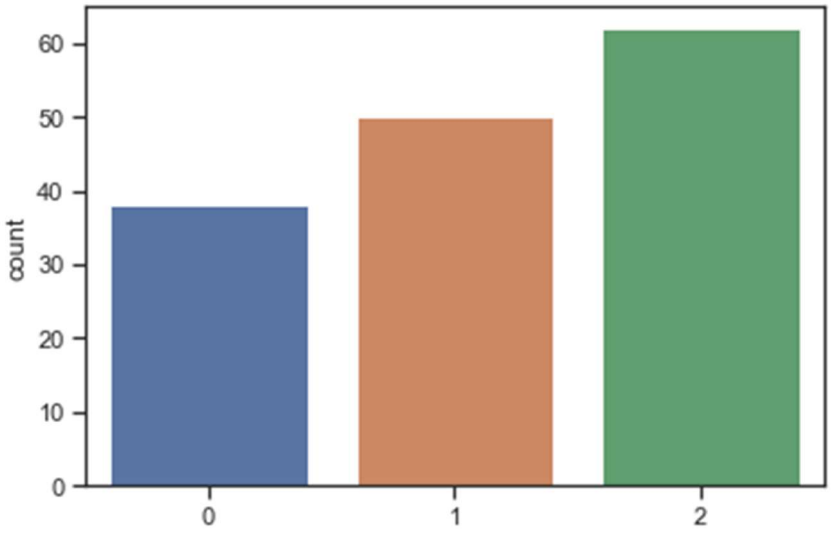
In [26]:

```
# k-means++ clustering http://scikit-
learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
k=3
kmPP = cluster.KMeans(n_clusters=k, init='k-means++', max_iter=300, tol=0.0001, n_jobs=-
1, random_state = random_state)
kmPP.fit(X)
C_kmpp = kmPP.predict(X)

sns.countplot(C_kmpp)
C_kmpp[:10]
```

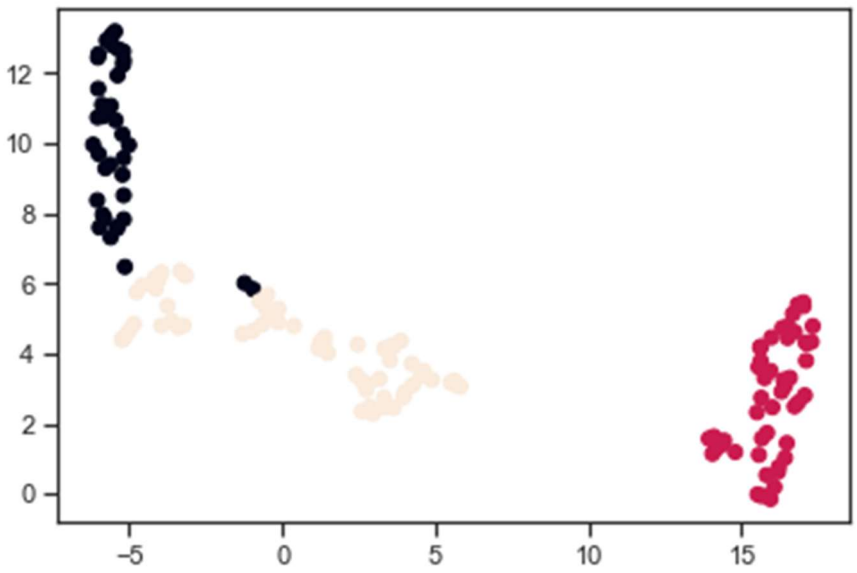
Out[26]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```



In [27]:

```
fig, ax = plt.subplots()
ax.scatter(X2D[:,0], X2D[:,1], c=C_kmpp)
plt.show()
```



Handling "Large Data" : Mini-Batch k-Means

Given: k, mini-batch size b, iterations t, data set X
Initialize each c ∈ C with an x picked randomly from X
v ← 0
for i = 1 to t do
    M ← b examples picked randomly from X
    for x ∈ M do
        d[x] ← f (C, x) // Cache the center nearest to x
    end for
    for x ∈ M do
        c ← d[x] // Get cached center for this x
        v[c] ← v[c] + 1 // Update per-center counts
        η ← 1 / v[c] // Get per-center learning rate
        c ← (1 – η)c + ηx // Take gradient step
    end for
end for

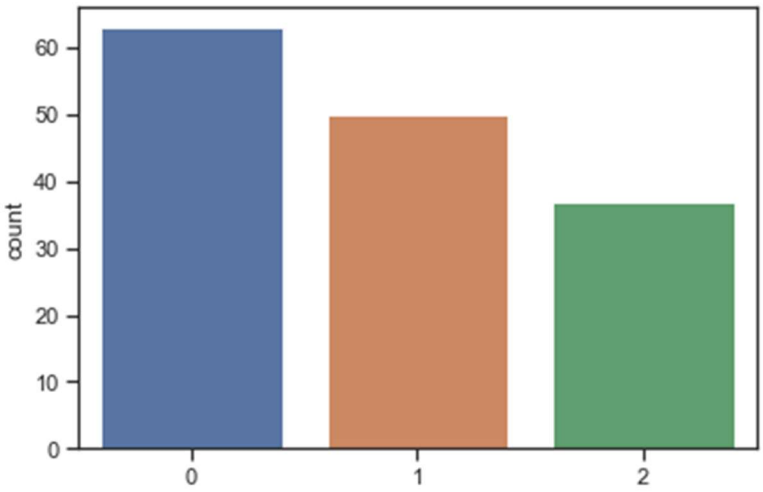
- **Referensi**: *Sculley, D. (2010, April). Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web* (pp. 1177-1178). ACM.
- Google

In [28]:

```
# MiniBatch k-Means
# http://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html
# minibatch "tidak bisa parallel"!!!...
# parameter penting km = batch_size ... pada aplikasi sesungguhnya disarankan "minimal"
3xk
mbkm = cluster.MiniBatchKMeans(n_clusters=k, init='random', max_iter=300, tol=0.0001,
batch_size = 100, random_state = random_state)
mbkm.fit(X)
C_mbkm = mbkm.predict(X)
sns.countplot(C_mbkm)
C_mbkm[:10]
```

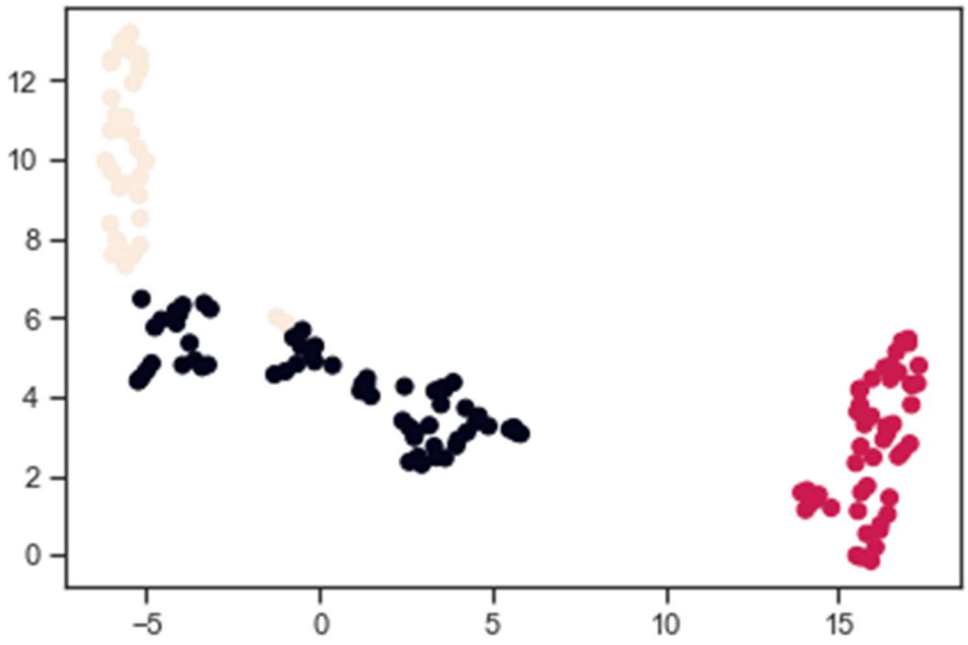
Out[28]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```



In [29]:

```
fig, ax = plt.subplots()
ax.scatter(X2D[:,0], X2D[:,1], c=C_mbkm)
plt.show()
```



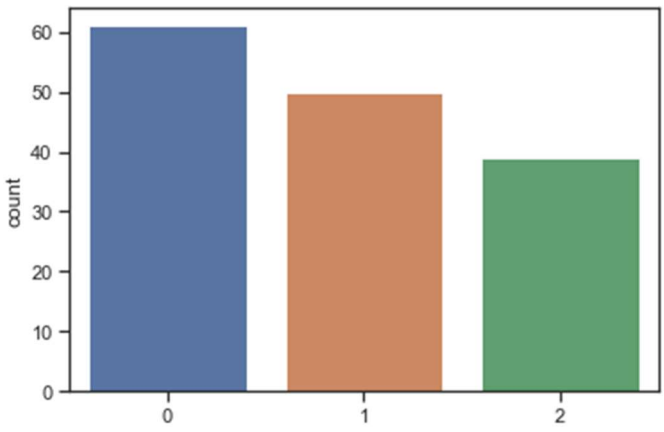
Minibatch k-Means++

In [30]:

```
# MiniBatch k-Means++
mbkmPP = cluster.MiniBatchKMeans(n_clusters=k, init='k-means++', max_iter=300,
tol=0.0001, random_state = random_state)
mbkmPP.fit(X)
C_mbkmPP = mbkmPP.predict(X)
sns.countplot(C_mbkmPP)
C_mbkmPP[:10]
```

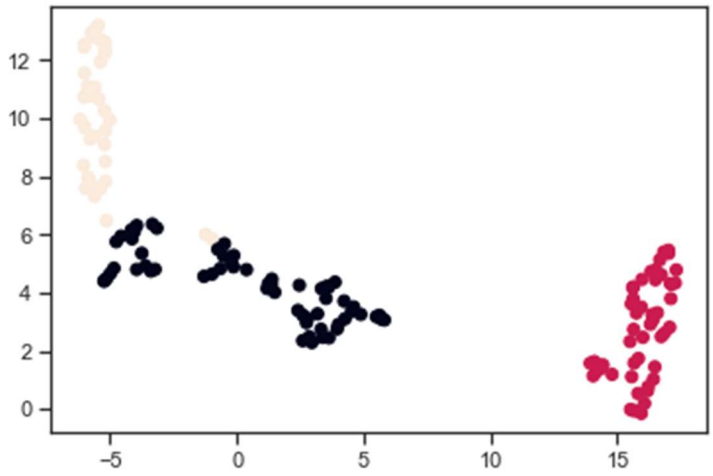
Out[30]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```



In [31]:

```
fig, ax = plt.subplots()
ax.scatter(X2D[:,0], X2D[:,1], c=C_mbkmPP)
plt.show()
```

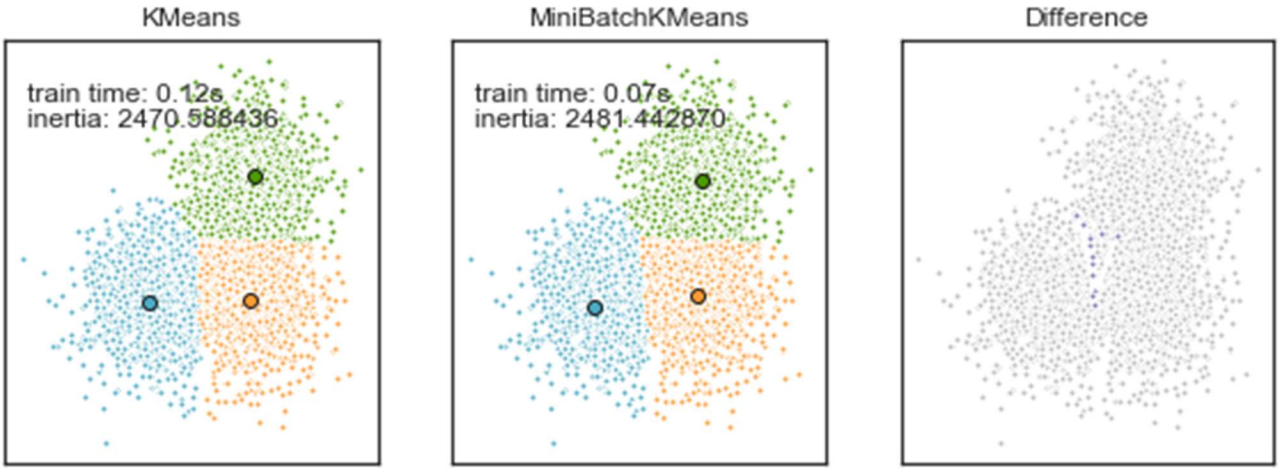


k_means VS MiniBatch k-Means?

In [32]:

```
tau.km_vs_mbkm()

# Ingat tujuan clustering analysis di Data Mining/Data Science
```



Evaluasi dan interpretasi k-Means: Silhouette Coefficient

The **silhouette coefficient** (SC) is an internal metric that measures compactness and separation of clusters at the same time. SC calculates the average distances within a cluster and minimum distance between an object to other clusters as follows:

$$SC = \frac{1}{N}\sum_{i=1}^{N}\frac{\beta_i - \alpha_i}{\max\{\alpha_i, \beta_i\}},$$

where α_i is the average distances of objects in a cluster, that is

$$\alpha_i = \frac{\sum_{j \neq i, x_j \in c_i} |x_i - x_j|}{|c_i|}$$

and β_i is the distance between an object x_i with the closest cluster centroid w_j . β_i is calculated as follows:

$$\beta_i = min\{|x_i - w_j|, j = 1,2, \dots, k, j \neq i\}.$$

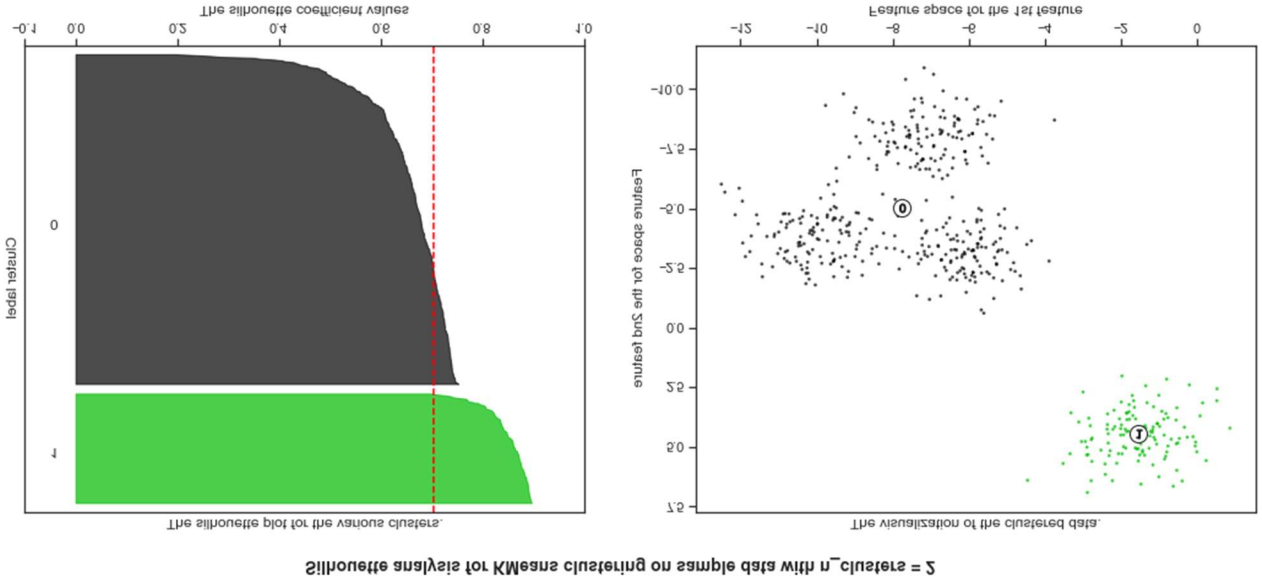
An SC value can range from 1 to -1 ($-1 \leq SC \leq 1$), where 1 means the clustering solution is "*correct*" and -1 means the clustering solution is "*incorrect*".

Apa makna intuitifnya?

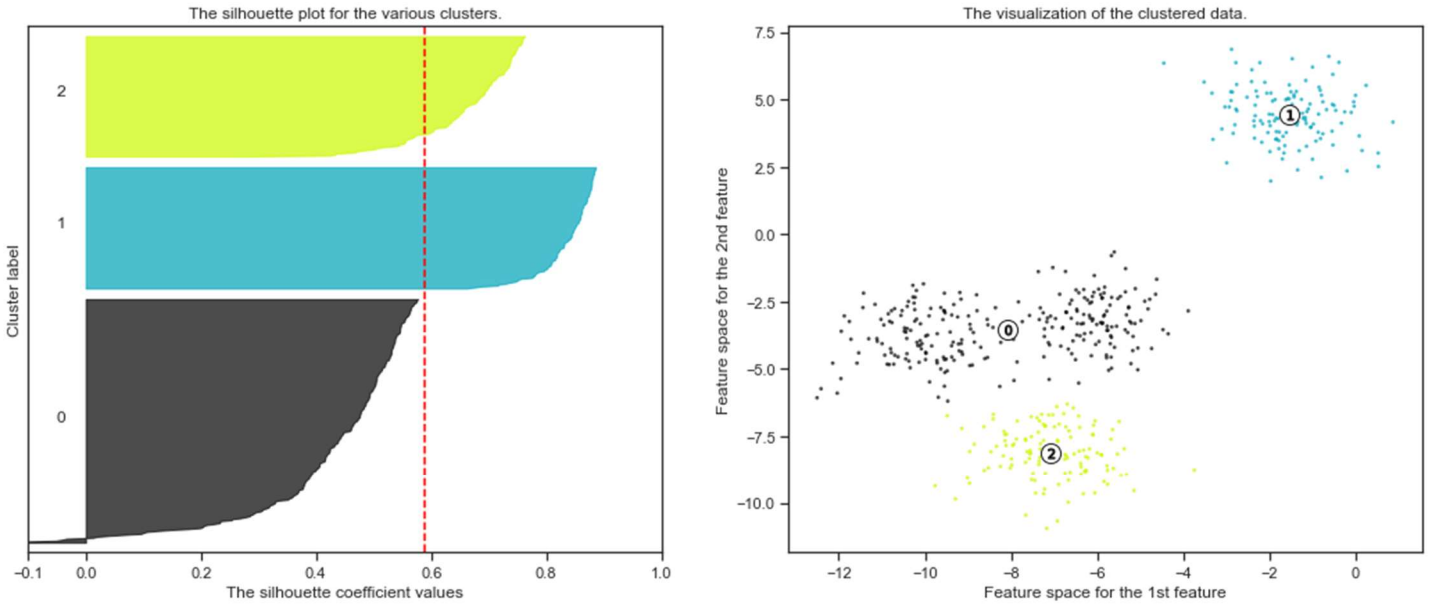
In [33]:

```
tau.sil_based_optimal_km()
```

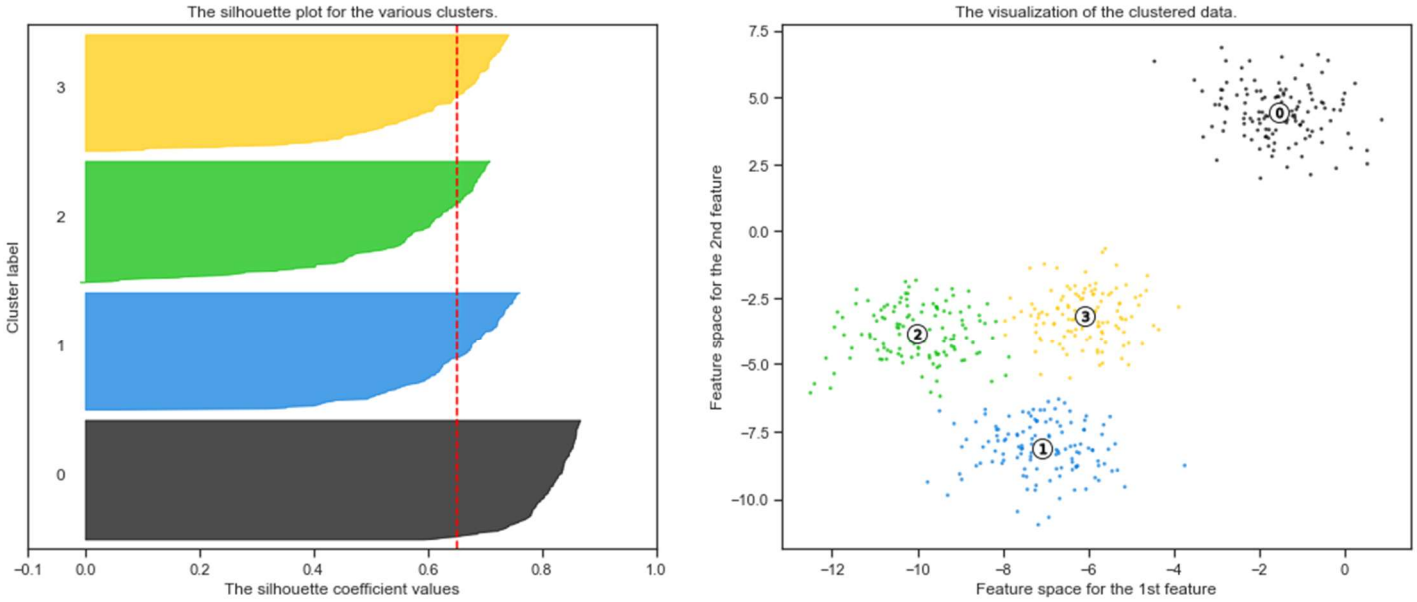
```
For n_clusters = 2 The average silhouette_score is : 0.7049787496083262
For n_clusters = 3 The average silhouette_score is : 0.5882004012129721
For n_clusters = 4 The average silhouette_score is : 0.6505186632729437
For n_clusters = 5 The average silhouette_score is : 0.5745566973301872
For n_clusters = 6 The average silhouette_score is : 0.4387644975296138
```

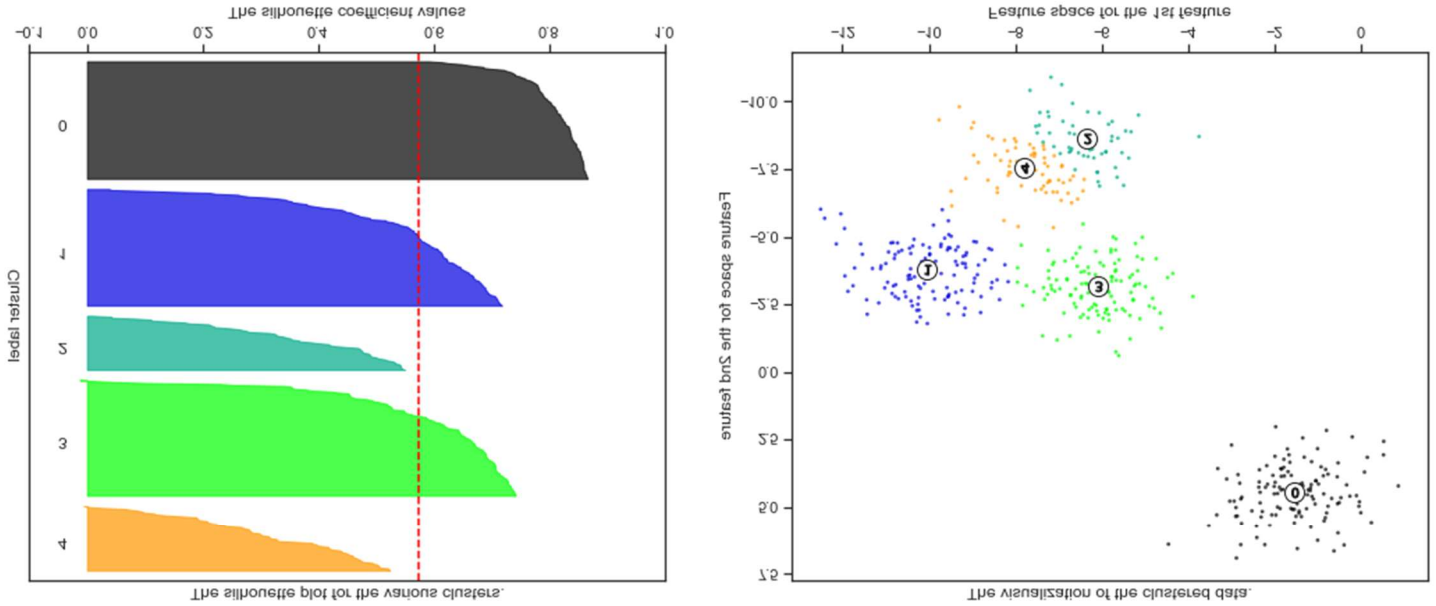


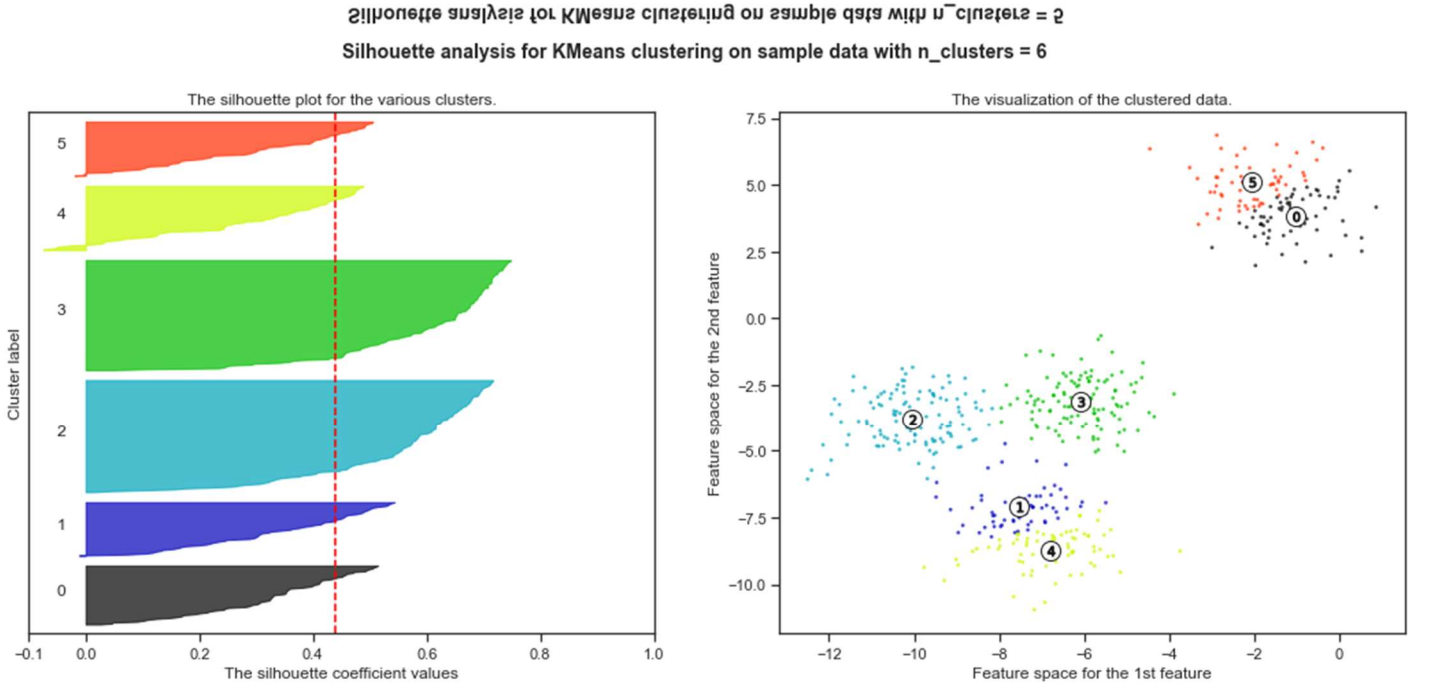
Silhouette analysis for KMeans clustering on sample data with n_clusters = 3



Silhouette analysis for KMeans clustering on sample data with n_clusters = 4





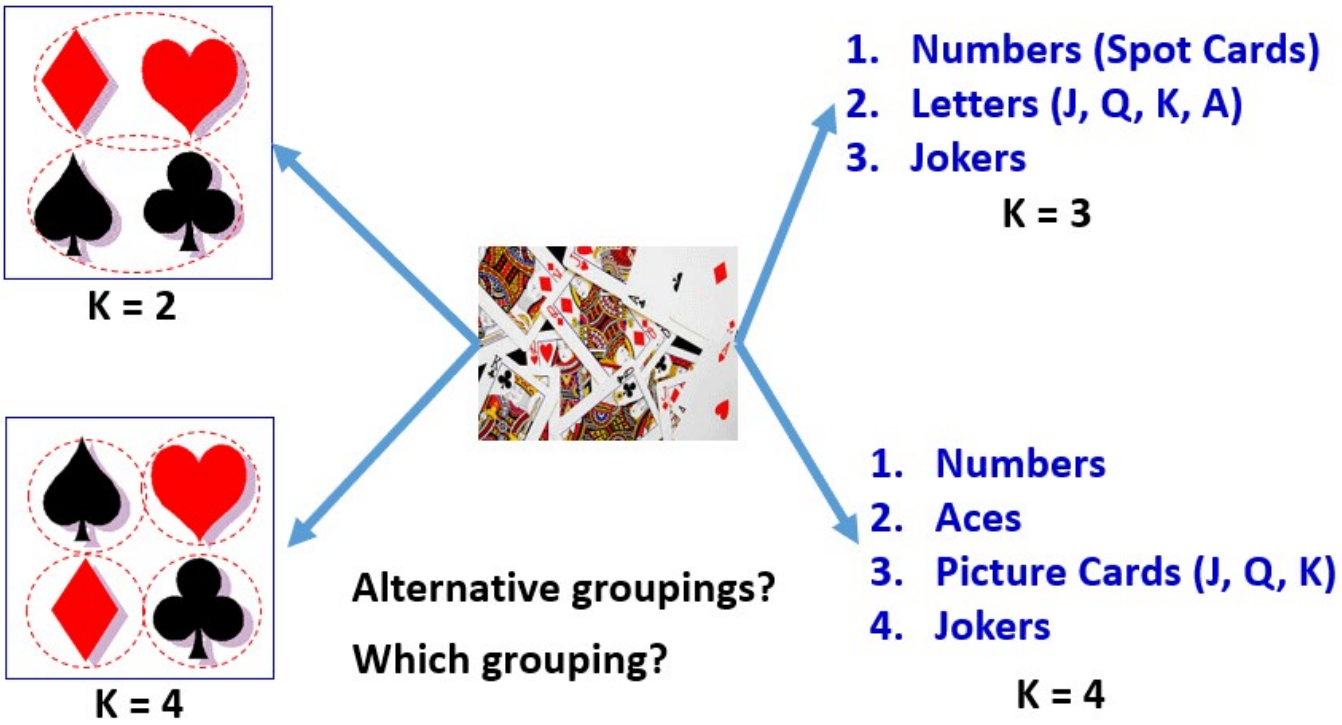


In [34]:

```
#Evaluasi : Internal . Contoh Silouette Coefficient ==> warning hanya cocok untuk k-means (centroid-based clustering)
Hasil_Clustering = [C_km, C_kmpp, C_mbkm, C_mbkmPP]
for res in Hasil_Clustering:
    print(siluet(X,res), end=', ')
# Bagaimana cara kerja dan interpretasinya?
```

```
0.5528190123564102, 0.5528190123564102, 0.5533422301756089, 0.5511916046195927,
```

Clustering?

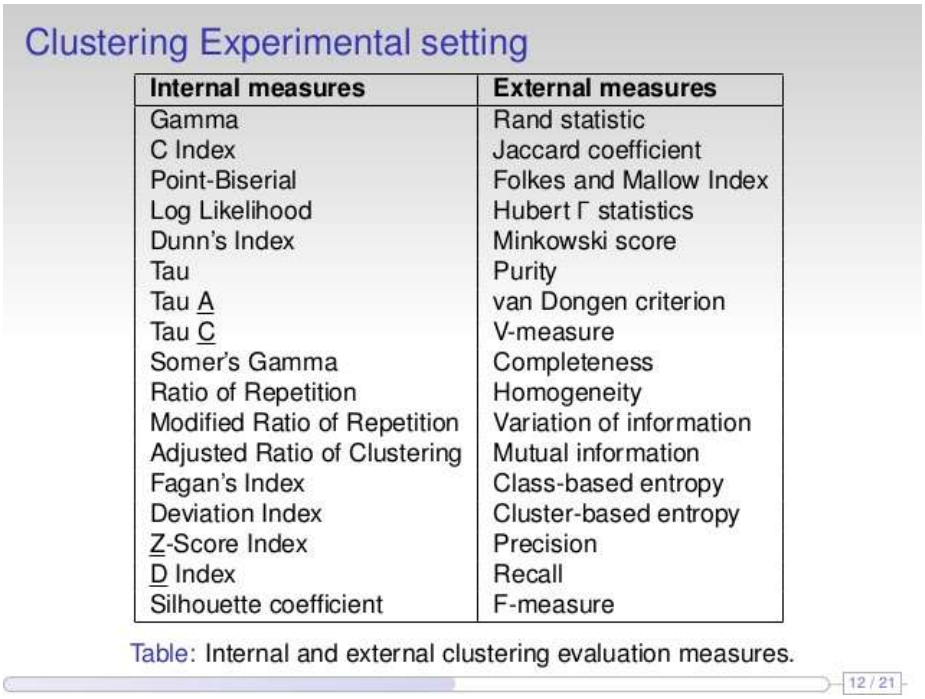


Tidak ada "Ground Truth" di Unsupervised Learning/Clustering.
Salah satu "Bias" terbesar adalah algoritma clustering yang kita pilih.

Catatan Penting dalam mengevaluasi Clustering secara internal:

- Tidak ada clustering yang "benar"
- Yang terpenting adalah interpretability/Informasi yang didapatkan (non-trivial information)
- Internal metric tertentu hanya cocok untuk suatu algoritma tertentu juga, sehingga di Penelitian/Aplikasi di dunia professional jangan membandingkan 2 macam clustering dengan ukuran internal yang spesifik untuk metode clustering tertentu (misal Silhouette untuk k-Means).
- Kleinberg, J. M. (2003). An impossibility theorem for clustering. In Advances in neural information processing systems (pp. 463-470).
- Referensi 1: http://papers.nips.cc/paper/2340-an-impossibility-theorem-for-clustering.pdf
- Referensi 2: https://core.ac.uk/download/pdf/34638775.pdf

Evaluasi Clustering: Internal VS External



Clustering Experimental setting

Internal measures	External measures
Gamma	Rand statistic
C Index	Jaccard coefficient
Point-Biserial	Folkes and Mallow Index
Log Likelihood	Hubert Γ statistics
Dunn's Index	Minkowski score
Tau	Purity
Tau A	van Dongen criterion
Tau C	V-measure
Somer's Gamma	Completeness
Ratio of Repetition	Homogeneity
Modified Ratio of Repetition	Variation of information
Adjusted Ratio of Clustering	Mutual information
Fagan's Index	Class-based entropy
Deviation Index	Cluster-based entropy
Z-Score Index	Precision
D Index	Recall
Silhouette coefficient	F-measure

Table: Internal and external clustering evaluation measures.

12 / 21

Di dunia akademis hanya diperkenalkan 2 macam evaluasi, tapi di aplikasi (industri) ada cara evaluasi ke-03.

In [35]:

```
# Bagaimana dengan evaluasi External?
# "C" adalah ground truth/golden standard
for res in Hasil_Clustering:
    print(purity(C,res), end=', ')
```

```
0.7514854021988338, 0.7514854021988338, 0.7429510367210881, 0.7364192881252849,
```

In [36]:

```
# Evaluasi External NMI
for res in Hasil_Clustering:
    print(NMI(C,res), end=', ')
# untuk F-Score ada juga code dan penjelasannya di blog post di atas
```

```
0.7581756800057784, 0.7581756800057784, 0.7507393176128085, 0.7419116631817836,
```

Please read more here: https://tau-data.id/evaluasi-eksternal/

Cara menarik kesimpulan dari k-Means: Interpretasi

In [37]:

```
kmPP.cluster_centers_
```

Out[37]:

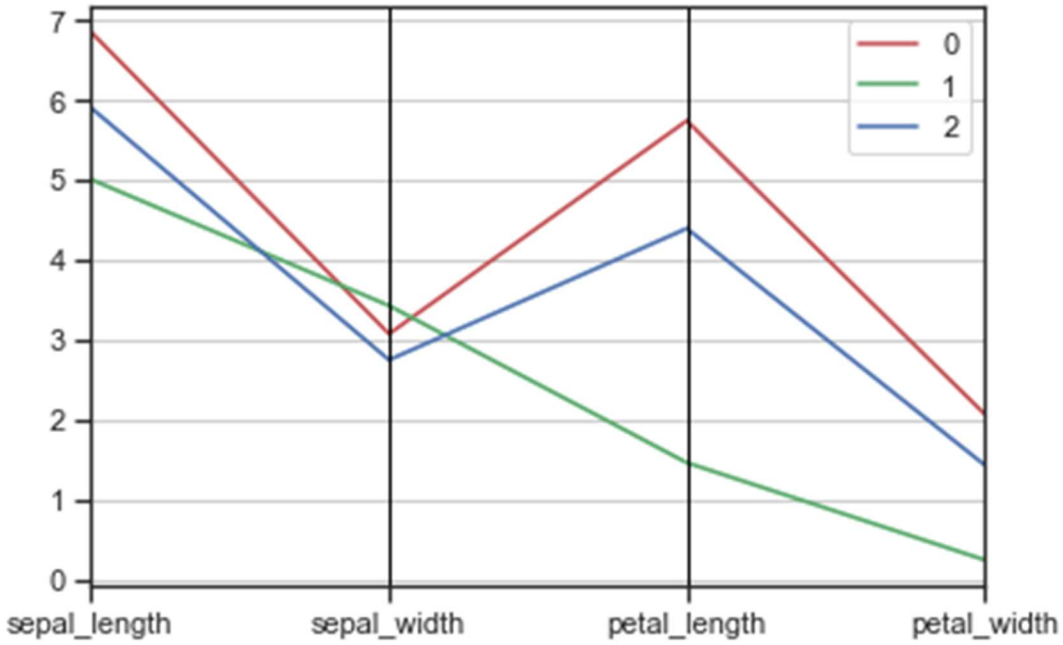
```
array([[6.85      , 3.07368421, 5.74210526, 2.07105263],
       [5.006     , 3.428     , 1.462     , 0.246     ],
       [5.9016129 , 2.7483871 , 4.39354839, 1.43387097]])
```

In [38]:

```
# Evaluasi sebenarnya tidak terlalu penting di Unsupervised learning.
# inilah yang membedakan "clustering" dan "clustering Analysis"
# yang lebih penting adalah interpretasi, tapi Bagaimana?
# contoh k-means++

cols = ['sepal_length','sepal_width','petal_length','petal_width']
dfC = pd.DataFrame(kmPP.cluster_centers_, columns=cols)
dfC['cluster'] = dfC.index

pd.plotting.parallel_coordinates(dfC, 'cluster', color=('r', 'g', 'b'))
plt.show()
```



In [39]:

```
kmPP.cluster_centers_
```

Out[39]:

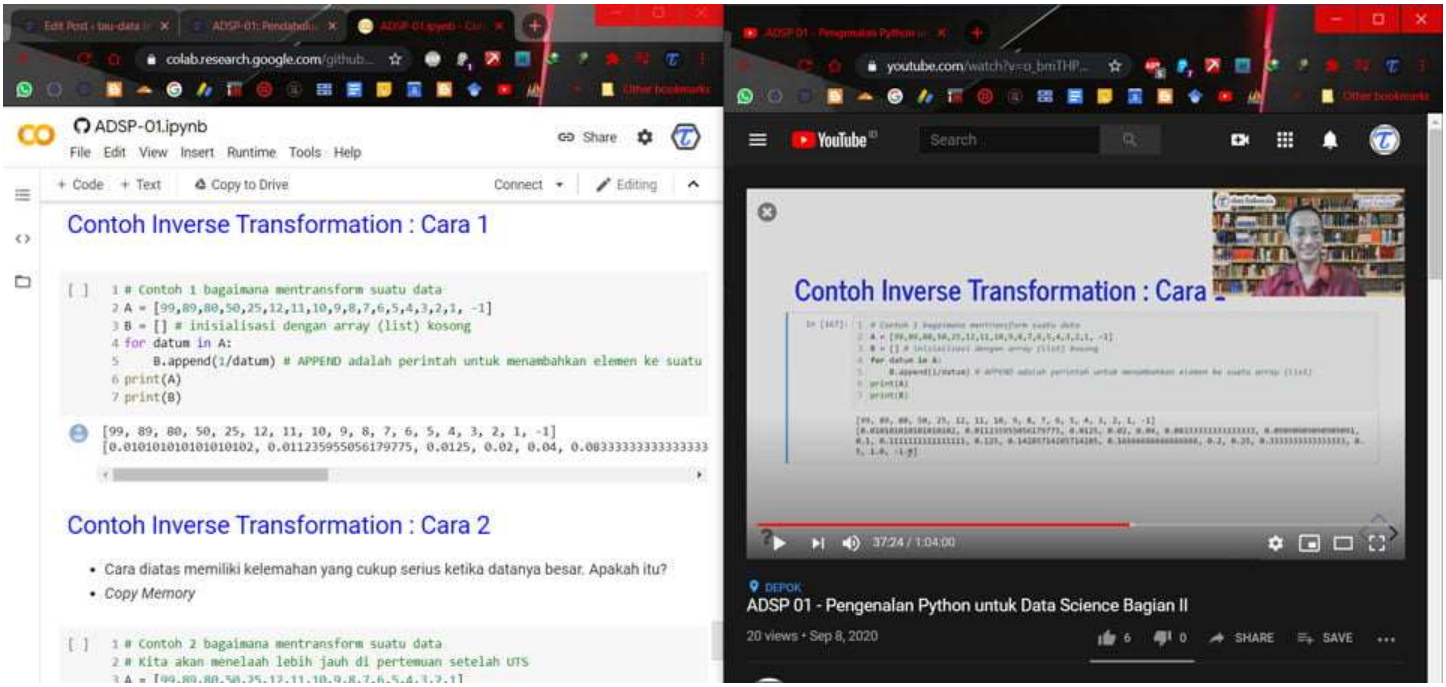
```
array([[6.85      , 3.07368421, 5.74210526, 2.07105263],
       [5.006     , 3.428     , 1.462     , 0.246     ],
       [5.9016129 , 2.7483871 , 4.39354839, 1.43387097]])
```

k-Means Best Practices

- Hati-hati faktor skala data ==> Normalisai/Standardized.
- Hati-hati asumsi topologi data di k-means.
- k-Means tidak bisa untuk data Kategori
- Sangat tidak disarankan untuk data tidak terstruktur berskala besar. Kalau datanya tidak besar cukup ganti jarak euclid dengan similarity Cosine.

Code Lesson EDA-03

Code dari lesson ini dapat di akses di Link berikut (wajib login ke Google/Gmail): **Code EDA-03**
Di link tersebut anda langsung bisa merubah code dan menjalankannya. Keterangan lebih lanjut di video yang disertakan. Sangat disarankan untuk membuka code dan video "**side-by-side**" untuk mendapatkan pengalaman belajar yang baik (Gambar dibawah). SIlahkan modifikasi (coba-coba) hal lain, selain yang ditunjukkan di video untuk mendapatkan pengalaman belajar yang lebih mendalam. Tentu saja juga silahkan akses berbagai referensi lain untuk memperkaya pengetahuan lalu diskusikan di forum yang telah disediakan.



"**Side-by-Side**": Ilustrasi bagaimana menggunakan code dan video dalam pembelajaran di tau-data. untuk mendapatkan pengalaman belajar yang baik.

Module/Code EDA-03: Pendahuluan Clustering

Referensi

1. Everitt, B. S., Landau, S., & Leese, M. (1993). Cluster analysis. 1993. *Edward Arnold and Halsted Press,*.
2. Arthur, D., & Vassilvitskii, S. (2006). *k-means++: The advantages of careful seeding*. Stanford.
3. Sculley, D. (2010, April). Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web* (pp. 1177-1178).
4. Jain, A.K., Data clustering: 50 years beyond K-means. Pattern Recognition Letters, 2010. 31(8): p. 651-666.
5. Pang-Ning, T., M. Steinbach, and V. Kumar, Introduction to data mining. Vol. 74. 2006, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.
6. Kleinberg, J. M. (2003). An impossibility theorem for clustering. In Advances in neural information processing systems (pp. 463-470).