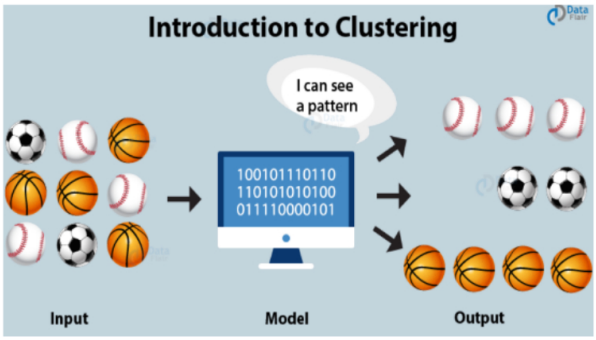
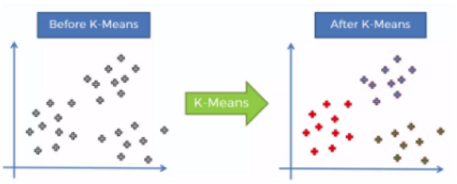
Week 11 - K Means Clustering

Konsep Clustering

Clustering adalah sebuah proses untuk mengelompokan data ke dalam beberapa cluster atau kelompok sehingga data dalam satu cluster memiliki tingkat kemiripan yang maksimum dan data antar cluster memiliki kemiripan yang minimum.



Metode K-means merupakan metode clustering yang paling sederhana dan umum. Hal ini dikarenakan K-means mempunyai kemampuan mengelompokkan data dalam jumlah yang cukup besar dengan waktu komputasi yang cepat dan efisien.



Cara kerja algoritma K-Means Clustering sebagai berikut :

Langkah 1: Tentukan berapa banyak cluster k dari dataset yang akan dibagi.

Langkah 2: Tetapkan secara acak data k menjadi pusat awal lokasi klaster.

Langkah 3: Untuk masing-masing data, temukan pusat cluster terdekat. Dengan demikian berarti masing-masing pusat cluster memiliki sebuah subset dari dataset, sehingga mewakili bagian dari dataset. Oleh karena itu, telah terbentuk cluster k: C1, C2, C3, ..., Ck .

Langkah 4: Untuk masing-masing cluster k, temukan pusat luasan klaster, dan perbarui lokasi dari masing-masing pusat cluster ke nilai baru dari pusat luasan.

Langkah 5: Ulangi langkah ke-3 dan ke-5 hingga data-data pada tiap cluster menjadi terpusat atau selesai.

Algoritma K-Means Clustering memiliki kelebihan dan juga kekurangan.

Kelebihan :

- Relatif sederhana dan mudah untuk diterapkan.
- Dapat diskalakan untuk dataset dalam jumlah besar.
- Mudah beradaptasi dengan contoh baru.
- Umum diimplementasikan ke cluster dengan bentuk dan ukuran yang berbeda, seperti cluster elips.

Kekurangan :

- Perlu menentukan nilai k secara manual
- Sangat bergantung pada inisialisasi awal. Jika nilai random untuk inisialisasi kurang baik, maka pengelompokkan yang dihasilkan pun menjadi kurang optimal.
- Dapat terjadi curse of dimensionality. Masalah ini timbul jika dataset memiliki dimensi yang sangat tinggi
- K-means mengalami kesulitan mengelompokkan data di mana cluster memiliki ukuran dan kepadatan yang bervariasi.

Implementasi 1 - Basic Clustering - Data Dummy Make Blobs

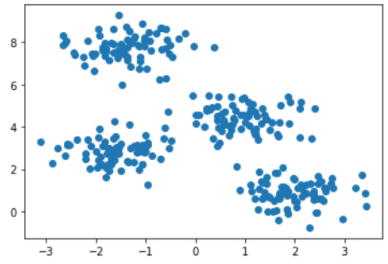
1. Import Library

```
[ ] import numpy as np #used for working with arrays and maths
    import pandas as pd #dataframe
    from matplotlib import pyplot as plt #plotting
    from sklearn.datasets import make_blobs #generate data
    from sklearn.cluster import KMeans #main algorithm

    #determining number of clusters
    from sklearn.metrics import silhouette_score
```

2. Load Dataset

```
X, y = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0) #make_blobs() adalah dataset yang dapat dimodelkan oleh K-Means clustering.
# n_samples : jumlah data
# centers : jumlah cluster
# cluster_std : std cluster

plt.scatter(X[:,0], X[:,1])
```

```
<matplotlib.collections.PathCollection at 0x7f0290de0250>
```



3. Determine Number of Clusters

Bagian ini diperlukan untuk menentukan jumlah cluster optimal, pada kasus kali ini jumlah cluster telah diketahui yaitu 4. Namun cara ini cocok untuk diterapkan pada dataset dengan sebaran yang berdekatan.

Method 1 - Elbow Method

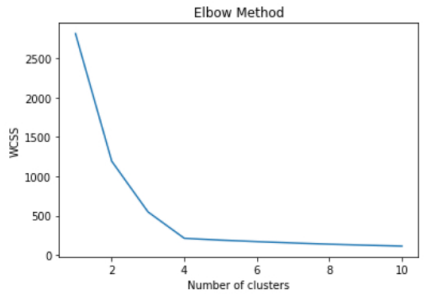
Metode Elbow merupakan suatu metode yang digunakan untuk menghasilkan informasi dalam menentukan jumlah cluster terbaik dengan cara melihat persentase hasil perbandingan antara jumlah cluster yang akan membentuk siku pada suatu titik.

Jika nilai cluster pertama dengan nilai cluster kedua memberikan sudut dalam grafik atau nilainya mengalami penurunan paling besar maka jumlah nilai cluster tersebut yang tepat. Untuk mendapatkan perbandingannya adalah dengan menghitung Sum of Square Error (SSE) dari masing-masing nilai cluster. Karena semakin besar jumlah nilai cluster K, maka nilai SSE akan semakin kecil.

Persamaan SSE = $$SSE = \sum_{k=1}^{K} \sum_{x_i} |x_i - c_k|^2$$

Keterangan:
K = *cluster* ke-*c*
x_i = jarak data obyek ke-*i*
c_k = pusat *cluster* ke-*i*

```
[ ] wcss = []
    for i in range(1, 11):
        kmeans = KMeans(n_clusters=i, init='k-means++',
                        max_iter=300, n_init=10, random_state=0)
        kmeans.fit(X)
        wcss.append(kmeans.inertia_)
    plt.plot(range(1, 11), wcss)
    plt.title('Elbow Method')
    plt.xlabel('Number of clusters')
    plt.ylabel('WCSS')
    plt.show()
```



Method 2 - Silhouette Coefficient

Silhouette Coefficient digunakan untuk melihat kualitas dan kekuatan cluster, seberapa baik atau buruknya suatu obyek ditempatkan dalam suatu cluster. Metode ini merupakan gabungan dari metode separasi dan kohesi.

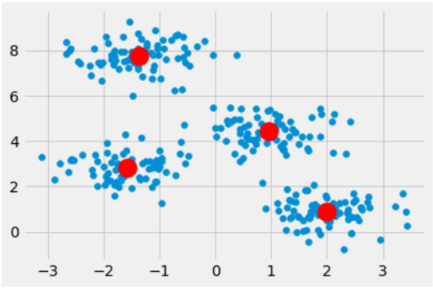
```
[ ] # A list holds the silhouette coefficients for each k
    silhouette_coefficients = []

    # Notice you start at 2 clusters for silhouette coefficient
    for k in range(2, 11):
        kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=300,
                        n_init=10, random_state=0)
```

```
        kmeans.fit(X)
        score = silhouette_score(X, kmeans.labels_)
        silhouette_coefficients.append(score)
```

```
plt.style.use("fivethirtyeight")
plt.plot(range(2, 11), silhouette_coefficients)
plt.xticks(range(2, 11))
plt.xlabel("Number of Clusters")
plt.ylabel("Silhouette Coefficient")
plt.show()
```

4. Create Cluster

```
[ ] kmeans = KMeans(n_clusters=4, init='k-means++', max_iter=300,
                    n_init=10, random_state=0)
    pred_y = kmeans.fit_predict(X)
    plt.scatter(X[:,0], X[:,1])
    plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
                s=300, c='red')
    plt.show()
```



Dari gambar diatas dapat kita lihat bahwa dari 300 data telah ter cluster menjadi 4.

Implementasi 2 - Mall Customer Segmentation

1. Import Library

```
import numpy as np
import pandas as pd

# Visualization packages
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
import warnings
warnings.filterwarnings('ignore')

# Determine number of clusters
from sklearn.metrics import silhouette_score
```

2. Load Dataset

```
[ ] df = pd.read_csv('/content/Mall_Customers.csv')
```

3. EDA

```
[ ] df.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6

3	4	Female	23	16	77
4	5	Female	31	17	40

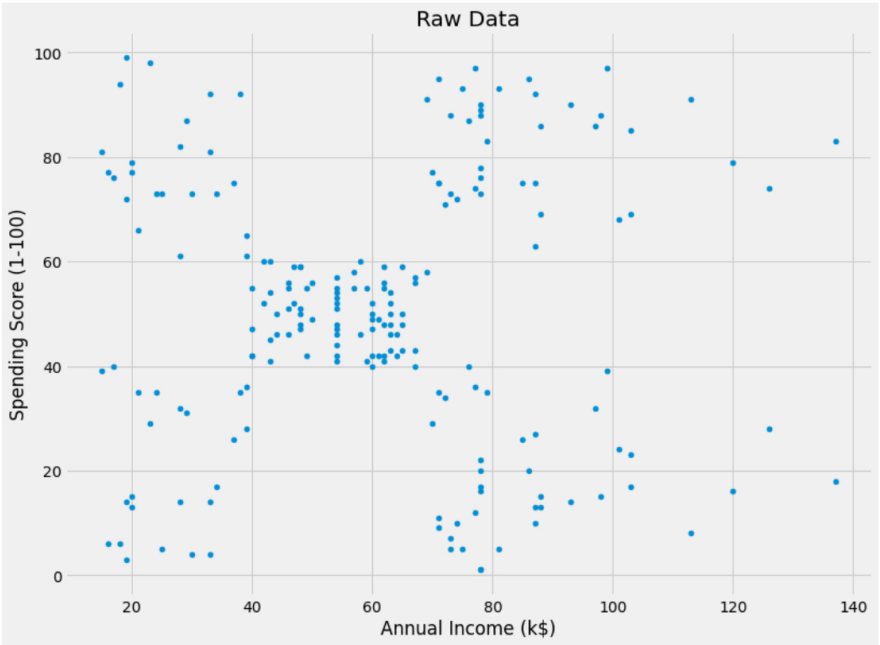
```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

```
[ ] df.describe()
```

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

```
[ ] plt.figure(figsize=(12,9))
    plt.scatter(df['Annual Income (k$)'], df['Spending Score (1-100)'], s = 25) #Point size is 25
    plt.title('Raw Data')
    plt.xlabel('Annual Income (k$)')
    plt.ylabel('Spending Score (1-100)')
    plt.show()
```



3. Data preprocessing

Select Feature

```
[ ] X = np.array(df.iloc[:,[3,4]])
```

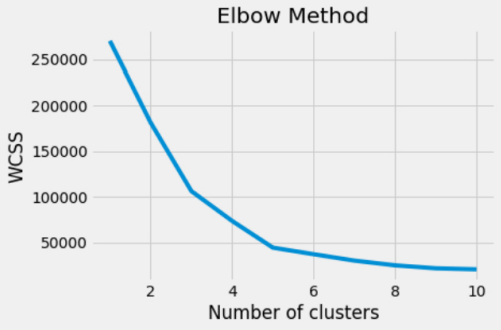
```
[ ] X
```

Determine Number of Clusters

Method 1 - Elbow Method

```
[ ] wcss = []
    for i in range(1, 11):
```

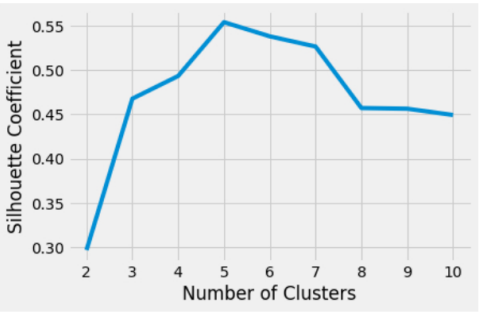
```
        kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
        kmeans.fit(X)
        wcss.append(kmeans.inertia_)
    plt.plot(range(1, 11), wcss)
    plt.title('Elbow Method')
    plt.xlabel('Number of clusters')
    plt.ylabel('WCSS')
    plt.show()
```



Method 2 - Silhouette Coefficient

```
[ ] # A list holds the silhouette coefficients for each k
    silhouette_coefficients = []

    # Notice you start at 2 clusters for silhouette coefficient
    for k in range(2, 11):
        kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=300, n_init=10, random_state=0)
        kmeans.fit(X)
        score = silhouette_score(X, kmeans.labels_)
        silhouette_coefficients.append(score)
```

```
    plt.style.use("fivethirtyeight")
    plt.plot(range(2, 11), silhouette_coefficients)
    plt.xticks(range(2, 11))
    plt.xlabel("Number of Clusters")
    plt.ylabel("Silhouette Coefficient")
    plt.show()
```



5. Create Cluster

```
[ ] kmeans = KMeans(n_clusters = 5, max_iter = 500, n_init = 10, random_state = 0)
    kmeans_preds = kmeans.fit_predict(X)
```

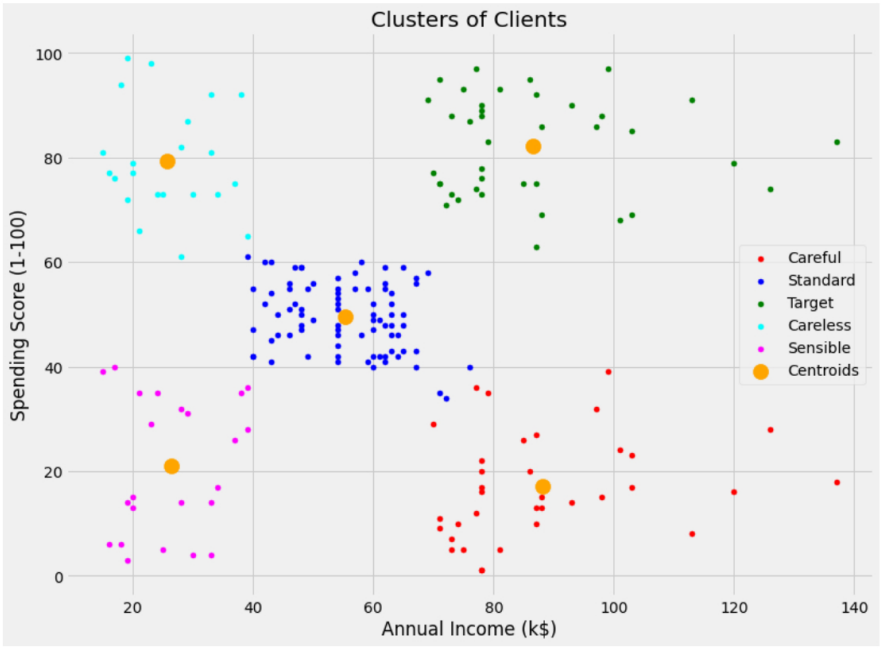
```
[ ] #predict result
    kmeans_preds
```

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

6. Show Cluster

```
[ ] point_size = 25
    colors = ['red', 'blue', 'green', 'cyan', 'magenta']
    labels = ['Careful', 'Standard', 'Target', 'Careless', 'Sensible']

    plt.figure(figsize = (12,9))
    for i in range(5):
        plt.scatter(X[kmeans_preds == i,0], X[kmeans_preds == i,1], s = point_size,
                    c = colors[i], label = labels[i])

    plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s = 200
```

```
            c = 'orange', label = 'Centroids')
plt.title('Clusters of Clients')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend(loc = 'best')
plt.show()
```



[ ]
```
#menambahkan hasil clustering ke dalam dataframe
df['Cluster'] = kmeans_preds
df.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Cluster
0	1	Male	19	15	39	4
1	2	Male	21	15	81	3
2	3	Female	20	16	6	4
3	4	Female	23	16	77	3
4	5	Female	31	17	40	4

[ ]
```
df.groupby(['Cluster'])['Cluster'].count()
```

```
Cluster
0    35
1    81
2    39
3    22
4    23
Name: Cluster, dtype: int64
```

[ ]
```
#menampilkan cluster tertentu
df.loc[df['Cluster'] == 1].head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Cluster
43	44	Female	31	39	61	1
46	47	Female	50	40	55	1
47	48	Female	27	40	47	1
48	49	Female	29	40	42	1
49	50	Female	31	40	42	1

[ ]
```
#predict one data
print(kmeans.predict([[100,20]]))
```

```
[0]
```

Implementasi 3 - Multi Features Clustering

Menggunakan dataset yang sama, **'Mall_Customers.csv'**, run ulang bagian **'Import Library'** dan **'Load Dataset'** pada **'Implementasi 2 - Mall Customer Segmentation'** apabila runtime habis

1. Import Library

```
[ ] #jalankan seluruh kode pada 'Implementasi 2 - Mall Customer Segmentation' apabila error karena runtime habis
    import plotly as py
    import plotly.graph_objs as go
```

3. Data preprocessing

Select Feature

```
[ ] X = np.array(df.iloc[:,[2,3,4]])
```

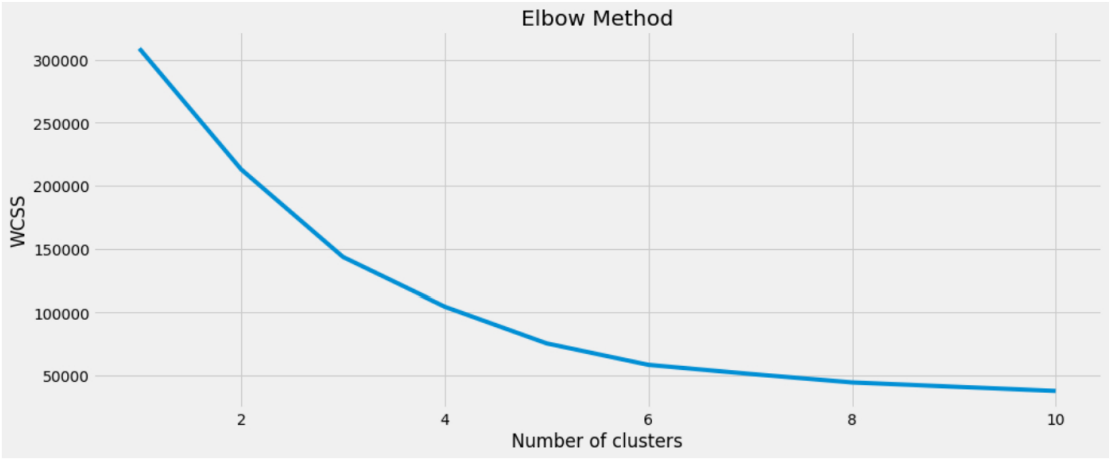
```
[ ] X[:5]
```

```
array([[19, 15, 39],
       [21, 15, 81],
       [20, 16,  6],
       [23, 16, 77],
       [31, 17, 40]])
```

Determine Number of Clusters

Method 1 - Elbow Method

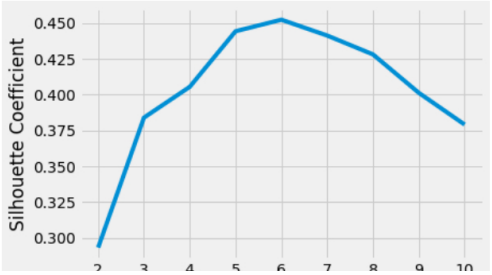
```
[ ] wcss = []
    for i in range(1, 11):
        kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
        kmeans.fit(X)
        wcss.append(kmeans.inertia_)
    plt.figure(1 , figsize = (15 ,6))
    plt.plot(range(1, 11), wcss)
    plt.title('Elbow Method')
    plt.xlabel('Number of clusters')
    plt.ylabel('WCSS')
    plt.show()
```



Method 2 - Silhouette Coefficient

```
[ ] # A list holds the silhouette coefficients for each k
    silhouette_coefficients = []

    # Notice you start at 2 clusters for silhouette coefficient
    for k in range(2, 11):
        kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=300, n_init=10, random_state=0)
        kmeans.fit(X)
        score = silhouette_score(X, kmeans.labels_)
        silhouette_coefficients.append(score)
```

```
    plt.style.use("fivethirtyeight")
    plt.plot(range(2, 11), silhouette_coefficients)
    plt.xticks(range(2, 11))
    plt.xlabel("Number of Clusters")
    plt.ylabel("Silhouette Coefficient")
    plt.show()
```



Number of Clusters

5. Create Cluster

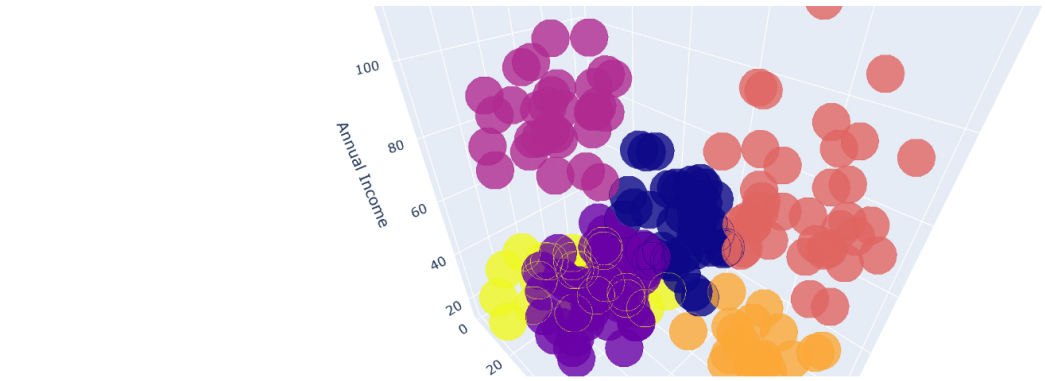
```
[ ] kmeans = KMeans(n_clusters = 6, max_iter = 500, n_init = 10, random_state = 0)
    kmeans_preds = kmeans.fit_predict(X)
```

```
[ ] #predict result
    kmeans_preds
```

```
array([5, 4, 5, 4, 5, 4, 5, 4, 5, 4, 5, 4, 5, 4, 5, 4, 5, 4, 5, 4,
       5, 4, 5, 4, 5, 4, 5, 4, 5, 4, 5, 4, 5, 4, 5, 4, 5, 4, 1, 4, 1, 0,
       5, 4, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0,
       1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0,
       0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 3, 0, 3, 2, 3, 2, 3, 2, 3,
       0, 3, 2, 3, 2, 3, 2, 3, 2, 3, 0, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3,
       2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3,
       2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3,
       2, 3], dtype=int32)
```

6. Show Cluster

```
[ ] labels3 = kmeans.labels_
    centroids3 = kmeans.cluster_centers_

    df['ClusterMultiFeatures'] =  labels3

    trace1 = go.Scatter3d(
        x= df['Age'],
        y= df['Spending Score (1-100)'],
        z= df['Annual Income (k$)'],
        mode='markers',
         marker=dict(
            color = df['ClusterMultiFeatures'],
            size= 20,
            line=dict(
                color= df['ClusterMultiFeatures'],
                width= 12
            ),
            opacity=0.8
         )
    )
    data = [trace1]
    layout = go.Layout(
        title= 'Clusters',
        scene = dict(
                xaxis = dict(title  = 'Age'),
                yaxis = dict(title  = 'Spending Score'),
                zaxis = dict(title  = 'Annual Income')
            )
    )
    fig = go.Figure(data=data, layout=layout)
    py.offline.iplot(fig)
```




```
[ ] df.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Cluster	ClusterMultiFeatures
0	1	Male	19	15	39	4	5
1	2	Male	21	15	81	3	4
2	3	Female	20	16	6	4	5

3	4	Female	23	16	77	3	4
4	5	Female	31	17	40	4	5

```
[ ] df.groupby(['ClusterMultiFeatures'])['ClusterMultiFeatures'].count()
```

```
ClusterMultiFeatures
0    38
1    45
2    35
3    39
4    22
5    21
Name: ClusterMultiFeatures, dtype: int64
```

```
[ ] #menampilkan cluster tertentu
    df.loc[df['ClusterMultiFeatures'] == 3].head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Cluster	ClusterMultiFeatures
123	124	Male	39	69	91	2	3
125	126	Female	31	70	77	2	3
127	128	Male	40	71	95	2	3
129	130	Male	38	71	75	2	3
131	132	Male	39	71	75	2	3

```
[ ] #predict one data
    print(kmeans.predict([[39,80,80]]))
```

```
[3]
```

Clustering Evaluation?

While Classification and Regression tasks form what's called Supervised Learning, Clustering forms the majority of Unsupervised Learning tasks. The difference between these two macro-areas lies in the type of data used. While in Supervised Learning samples are labelled with either a categorical label (Classification) or a numerical value (Regression), in Unsupervised Learning samples are not labelled, making it a relatively complex task to perform and evaluate.

Correctly measuring the performance of Clustering algorithms is key. This is especially true as it often happens that clusters are manually and qualitatively inspected to determine whether the results are meaningful.

Another main metrics used to evaluate the performance of Clustering algorithms are :

- Silhouette Score
- Rand Index
- Adjusted Rand Index
- Mutual Information
- Calinski-Harabasz Index
- Davies-Bouldin Index

TUGAS 😺

Lakukan Clustering dengan menggunakan dataset iris dari sklearn.datasets (dapat dilihat pada kode di bawah), gunakan referensi 'Implementasi 2 dan 3' di atas kemudian sesuaikan kodenya

Hasil model clustering dalam bentuk plot scatter dan kolom baru dengan nama 'cluster'

```
[ ] #iris dataset
    from sklearn import datasets
    iris = datasets.load_iris()

    df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
    df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

References :

https://towardsdatascience.com/machine-learning-algorithms-part-9-k-means-example-in-python-f2ad05ed5203

https://realpython.com/k-means-clustering-python/

https://www.dqlab.id/k-means-clustering-salah-satu-contoh-teknik-analisis-data-populer

https://blog.rosihanari.net/tutorial-k-means-clustering-dengan-python

https://towardsdatascience.com/three-performance-evaluation-metrics-of-clustering-when-ground-truth-labels-are-not-available-ee08cb3ff4fb

https://towardsdatascience.com/performance-metrics-in-machine-learning-part-3-clustering-d69550662dc6

https://socs.binus.ac.id/2017/03/09/clustering/

https://www.dqlab.id/k-means-clustering-salah-satu-contoh-teknik-analisis-data-populer