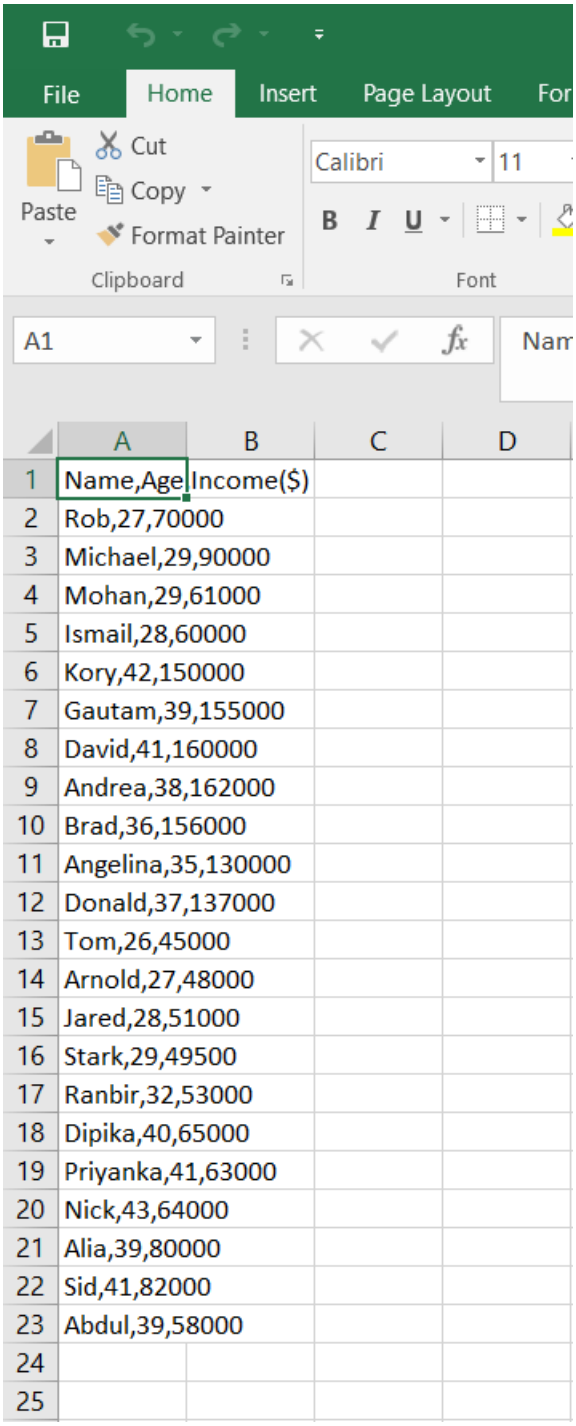
BÀI TOÁN PHÂN CỤM KHÁCH HÀNG CƠ BẢN

Dataset: Thông tin về Tên, tuổi và thu nhập của một số người



	A	B	C	D
1	Name,Age	Income($)		
2	Rob,27,70000			
3	Michael,29,90000			
4	Mohan,29,61000			
5	Ismail,28,60000			
6	Kory,42,150000			
7	Gautam,39,155000			
8	David,41,160000			
9	Andrea,38,162000			
10	Brad,36,156000			
11	Angelina,35,130000			
12	Donald,37,137000			
13	Tom,26,45000			
14	Arnold,27,48000			
15	Jared,28,51000			
16	Stark,29,49500			
17	Ranbir,32,53000			
18	Dipika,40,65000			
19	Priyanka,41,63000			
20	Nick,43,64000			
21	Alia,39,80000			
22	Sid,41,82000			
23	Abdul,39,58000			
24				
25				

Khai báo một số thư viện cần thiết:

```
In [2]: from sklearn.cluster import KMeans
        import pandas as pd
        from sklearn.preprocessing import MinMaxScaler
        from matplotlib import pyplot as plt
        %matplotlib inline
```

Đọc file income.csv:

```
In [6]: df = pd.read_csv("income.csv")
        df.head()
```

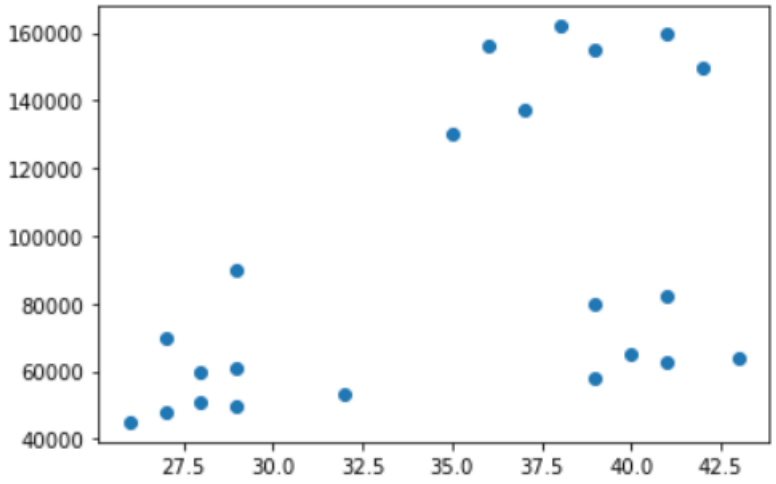
Out[6]:

	Name	Age	Income($)
0	Rob	27	70000
1	Michael	29	90000
2	Mohan	29	61000
3	Ismail	28	60000
4	Kory	42	150000

Vẽ đồ thị Scatter về 2 giá trị Age và Income($):

```
In [7]: plt.scatter(df['Age'], df["Income($)"])
```

```
Out[7]: <matplotlib.collections.PathCollection at 0x1ed1377bbb0>
```



Ta dễ dàng thấy dữ liệu dường như sẽ gồm 3 cụm.

Khai báo Kmeans với số lớp là 3:

```
In [9]: km = KMeans(n_clusters=3)
        print(km)

        KMeans(n_clusters=3)
```

Fit and predict:

```
In [10]: y_predicted = km.fit_predict(df[["Age", "Income($)"]])
         y_predicted

Out[10]: array([0, 0, 2, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 2])
```

Hiển thị kết quả:

```
In [19]: df["cluster"] = y_predicted
         df.head()
```

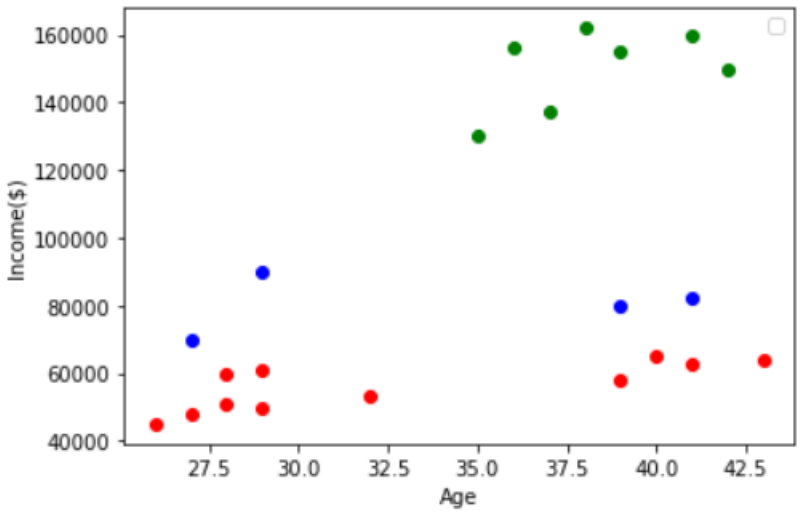
Out[19]:

	Name	Age	Income($)	cluster
0	Rob	27	70000	2
1	Michael	29	90000	2
2	Mohan	29	61000	0
3	Ismail	28	60000	0
4	Kory	42	150000	1

```
In [23]: df1 = df[df.cluster == 0]
         df2 = df[df.cluster == 1]
         df3 = df[df.cluster == 2]

         plt.scatter(df1["Age"], df1["Income($)"], color = "red")
         plt.scatter(df2["Age"], df2["Income($)"], color = "green")
         plt.scatter(df3["Age"], df3["Income($)"], color = "blue")

         plt.xlabel("Age")
         plt.ylabel("Income($)")
         plt.legend()
```

```
No handles with labels found to put in legend.
```

```
Out[23]: <matplotlib.legend.Legend at 0x1ed1500f490>
```



Có thể thấy cluster xanh bị phân nhóm chưa đúng!!

Nguyên nhân: scaling chưa đúng, trục Y (Income) từ 40000 → 160000, còn trục x được chia rất hẹp. Đó là lý do ta phải thực hiện việc tiền xử lý dữ liệu và sử dụng MinMaxScaler để scale những datapoint và sau đó mới thực hiện thuật toán Kmeans

scaler = MinMaxScaler()

scaler.fit(df[['Income($)']])

df['Income($)'] = scaler.transform(df[['Income($)']])

scaler.fit(df[['Age']])

df['Age'] = scaler.transform(df[['Age']])

```
In [26]: scaler = MinMaxScaler()

         scaler.fit(df[['Income($)']])
         df['Income($)'] = scaler.transform(df[['Income($)']])

         scaler.fit(df[['Age']])
         df['Age'] = scaler.transform(df[['Age']])
```

Đoạn code trên thực hiện Scale giá trị Income và Age vào trong khoảng [0, 1]

Có thể thấy kết quả lúc này ở cột Age và Income đã được Scale trong khoảng [0, 1]

	Name	Age	Income($)	cluster
0	Rob	0.058824	0.213675	2
1	Michael	0.176471	0.384615	2
2	Mohan	0.176471	0.136752	0
3	Ismail	0.117647	0.128205	0
4	Kory	0.941176	0.897436	1
5	Gautam	0.764706	0.940171	1
6	David	0.882353	0.982906	1
7	Andrea	0.705882	1.000000	1
8	Brad	0.588235	0.948718	1
9	Angelina	0.529412	0.726496	1
10	Donald	0.647059	0.786325	1
11	Tom	0.000000	0.000000	0
12	Arnold	0.058824	0.025641	0
13	Jared	0.117647	0.051282	0
14	Stark	0.176471	0.038462	0
15	Ranbir	0.352941	0.068376	0
16	Dipika	0.823529	0.170940	0
17	Priyanka	0.882353	0.153846	0
18	Nick	1.000000	0.162393	0
19	Alia	0.764706	0.299145	2
20	Sid	0.882353	0.316239	2
21	Abdul	0.764706	0.111111	0

Bây giờ thực hiện thuật toán Kmeans một lần nữa lên các điểm dữ liệu:

In [28]:
```
km = KMeans(n_clusters=3)
y_predicted = km.fit_predict(df[["Age", "Income($)"]])
y_predicted
```

Out[28]:
```
array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2])
```

In [29]:
```
df["cluster_scale"] = y_predicted
df.head()
```

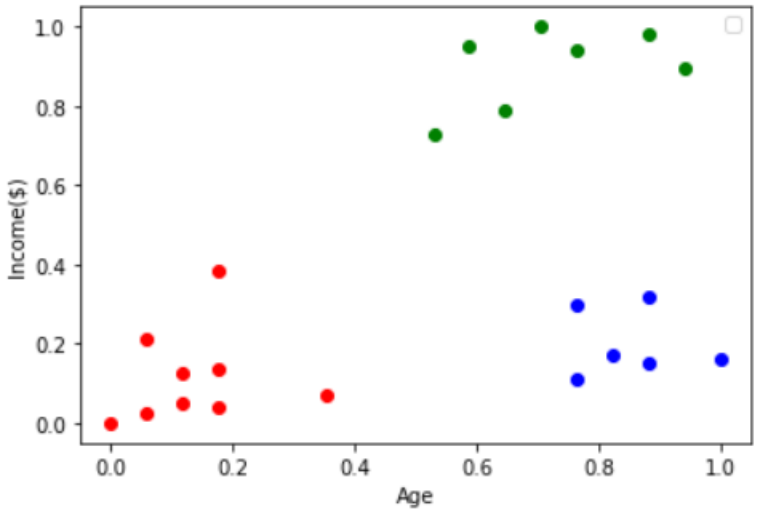
Out[29]:

	Name	Age	Income($)	cluster	cluster_scale
0	Rob	0.058824	0.213675	2	0
1	Michael	0.176471	0.384615	2	0
2	Mohan	0.176471	0.136752	0	0
3	Ismail	0.117647	0.128205	0	0
4	Kory	0.941176	0.897436	1	1

```
In [30]: df1 = df[df.cluster_scale == 0]
         df2 = df[df.cluster_scale == 1]
         df3 = df[df.cluster_scale == 2]

         plt.scatter(df1["Age"], df1["Income($)"], color = "red")
         plt.scatter(df2["Age"], df2["Income($)"], color = "green")
         plt.scatter(df3["Age"], df3["Income($)"], color = "blue")

         plt.xlabel("Age")
         plt.ylabel("Income($)")
         plt.legend()
```

```
No handles with labels found to put in legend.
```

```
Out[30]: <matplotlib.legend.Legend at 0x1ed15374af0>
```



Ta thấy kết quả phân cụm lúc này đã tốt hơn ban đầu rất nhiều.

Ta có thể kiểm tra tâm các cụm bằng lệnh:

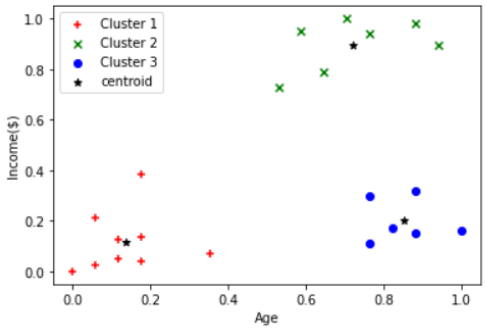
```
In [31]: km.cluster_centers_
```

```
Out[31]: array([[0.1372549 , 0.11633428],
                [0.72268908, 0.8974359 ],
                [0.85294118, 0.2022792 ]])
```

Ta có thể vẽ các tâm này vào các cụm:

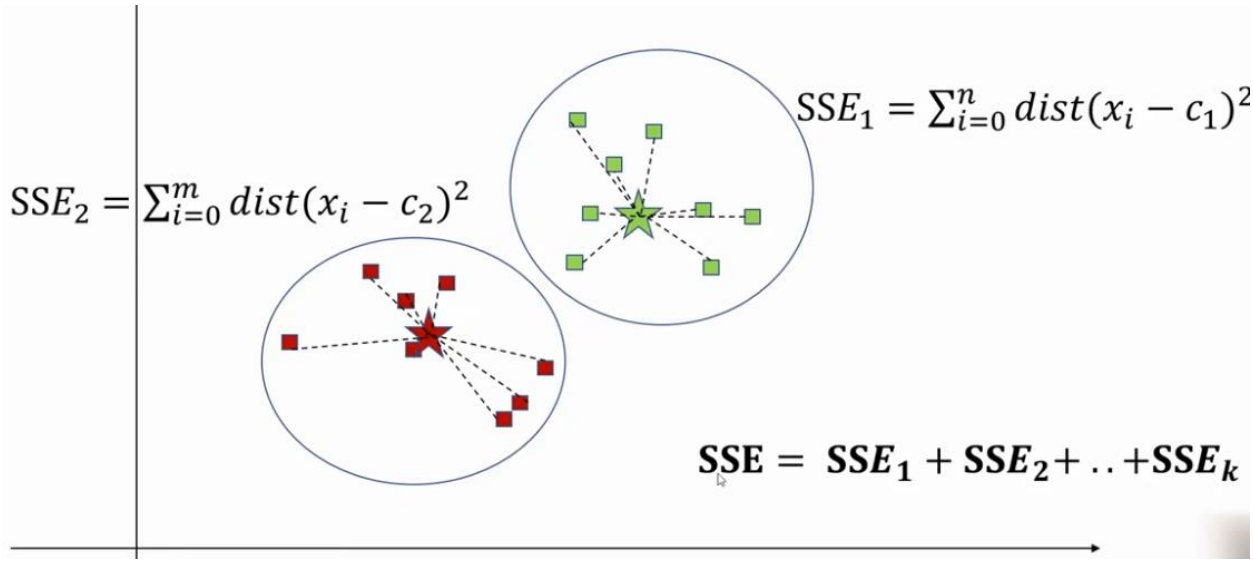
In [38]:
```
df1 = df[df.cluster_scale == 0]
df2 = df[df.cluster_scale == 1]
df3 = df[df.cluster_scale == 2]

plt.scatter(df1["Age"], df1["Income($)"], color = "red", marker = "+", label = "Cluster 1")
plt.scatter(df2["Age"], df2["Income($)"], color = "green", marker = "x", label = "Cluster 2")
plt.scatter(df3["Age"], df3["Income($)"], color = "blue", marker = "o", label = "Cluster 3")

# Plot Centers:
plt.scatter(km.cluster_centers_[:, 0], km.cluster_centers_[:, 1], color = "black", marker = "*", label = "centroid")
#plt.scatter(x, y), voi x la cot thu nhat (3 gia tri x_center) cua ma tran Center,
# --> x = km.cluster_centers_[:, 0]: duyệt qua tất cả các hàng, lấy cột đầu tiên (0)
# y la cot thu hai (3 gia tri y_center) cua ma tran Center
# --> y = km.cluster_centers_[:, 1]: duyệt qua tất cả các hàng, lấy cột thứ hai (1)

plt.xlabel("Age")
plt.ylabel("Income($)")
plt.legend()
```

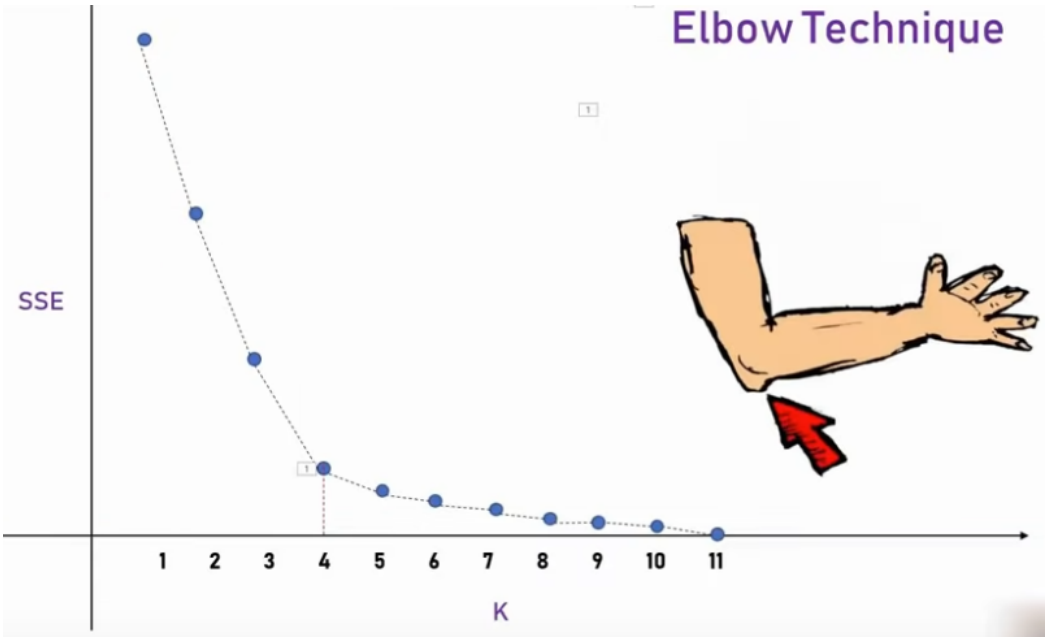
Out[38]: `<matplotlib.legend.Legend at 0x1ed1724dac0>`



Thuật toán Elbow:

Vì dataset ta đang sử dụng là khá ít và đơn giản, có thể vẽ ra để xem trước và dự đoán được bằng trực giác giá trị k dễ dàng. Tuy nhiên ở những dự án thực tế thì ta sẽ gặp những dataset có nhiều đặc trưng hơn (Ví dụ 20 đặc trưng) thì rất khó vẽ ra dữ liệu hiển thị để ta có thể quan sát và tìm ra giá trị k phù hợp bằng trực giác nữa. Do đó ta phải sử dụng thuật toán Elbow để tìm ra giá trị k phù hợp:





Ở mỗi vòng lập ứng với mỗi giá trị k, khi ta gọi hàm km.fit() thì có một thông số là inertia sẽ cho ta the sum of square error, ta chỉ cần thêm giá trị này vào mảng SSE ban đầu.

```
In [42]: k_rng = range(1, 10)
         sse = []
         for k in k_rng:
             km = KMeans(n_clusters=k)
             km.fit(df[["Age", "Income($)"]])
             sse.append(km.inertia_)

         C:\Users\admin\anaconda3\lib\site-pa
         n Windows with MKL, when there are l
         _NUM_THREADS=1.
           warnings.warn(

In [43]: sse

Out[43]: [5.434011511988179,
          2.091136388699078,
          0.4750783498553097,
          0.3491047094419566,
          0.2766936276300279,
          0.23218338957108078,
          0.1685851223602976,
          0.1422298983018973,
          0.10919063774844096]
```

Hiển thị lên đồ thị:

```
In [45]: plt.plot(k_rng, sse)

Out[45]: [<matplotlib.lines.Line2D at 0x1ed174898b0>]
```