CLUSTERING WITH K-MEANS DATA TRAIN

Nama: Chlaudiah Julinar
NIM : 1301150434
Kelas : IF 39 10

A. Library

Berikut adalah penjelasan mengenai library yang digunakan:

1. Numpy merupakan package yang paling dasar yang dibutuhkan untuk scientific computing, seperti digunakan untuk mendukung multidimensi array, berbagai jenis object seperti array, dll.
2. Pandas merupakan Python Data Analysis Toolkit. Dapat digunakan untuk melakukan read pada jenis tipe data apapun
3. Plotly digunakan untuk melakukan visualisasi data. Apabila pada python atau anaconda yang digunakan belum memiliki package plotly maka harus dilakukan install melalui pip terlebih dahulu

In [1]:
```
from copy import deepcopy
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
```

B. Load Data Train

Melakukan loading data pada file Train Set.csv dengan menggunakan package pandas. Tidak perlu menggunakan tab separated seperti delimiter karena pada file dengan tipe csv sudah secara otomatis mengabaikan tab. Setelah Train Set.csv sudah diload maka dapat dilihat bahwa terdapat 2 kolom yang artinya memiliki 2 data yaitu Value1, Value2. Dengan ukuran matriks yaitu 688 x 2.

In [2]:
```
file = pd.read_csv('Train Set.csv')
file
```

Out[2]:

	Value1	Value2
0	21.30	20.80
1	20.15	20.90
2	19.20	21.35
3	19.10	21.85
4	18.45	22.80
5	19.40	23.00
6	19.55	22.25
7	19.80	21.85
8	20.50	21.85
9	21.70	21.90
10	21.40	22.30
11	21.00	22.60
12	21.15	22.95
13	19.75	23.65
14	19.20	23.70
15	18.45	24.35

	Value1	Value2
16	20.65	23.85
17	19.70	24.60
18	20.15	25.05
19	22.15	25.10
20	21.60	24.65
21	21.90	23.65
22	22.55	23.50
23	22.55	24.30
24	23.30	24.45
25	24.25	24.35
26	23.80	25.25
27	23.40	23.80
28	22.90	23.20
29	22.30	22.80
...	...	...
658	34.40	25.60
659	7.80	13.70
660	8.85	13.35
661	9.00	12.70
662	8.05	12.90
663	7.70	13.25
664	6.80	13.20

	Value1	Value2
665	6.60	13.45
666	6.20	12.55
667	5.40	12.85
668	5.20	11.90
669	5.15	11.35
670	5.85	11.20
671	6.10	11.75
672	7.00	12.35
673	7.05	12.45
674	7.90	12.50
675	8.55	12.10
676	7.10	11.95
677	6.90	11.50
678	6.85	10.90
679	6.40	10.70
680	5.90	10.30
681	6.40	10.25
682	7.05	10.05
683	7.35	10.50
684	8.10	11.20
685	8.80	11.40
686	9.00	10.90

	Value1	Value2
687	9.35	10.50

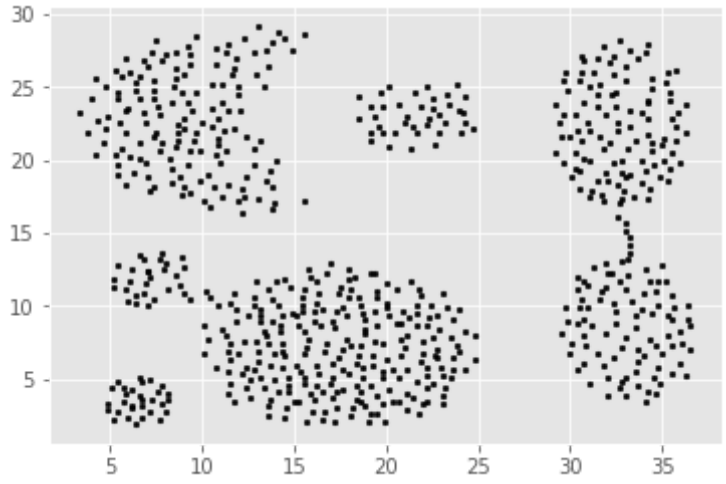
688 rows × 2 columns

C. Visualisasi Data

Karena terdapat 2 kolom, maka data dari kedua kolom di pisahkan terlebih dahulu dan ditampung ke dalam dua variabel yang berbeda. Kedua variablel tersebut, akan di plot kedalam sebuah scatter dengan semua warna data berwarna hitam yang artinya data tersebut masih merupakan seluruh data dibagian yang sama.

In [3]:
```
value1 = file['Value1'].values
value2 = file['Value2'].values
data = np.array(list(zip(value1, value2)))
plt.scatter(value1, value2, c='black', s=7)
```

Out[3]:
```
<matplotlib.collections.PathCollection at 0xf04c0d0>
```



D. Clustering Data Train

D.1 Fungsi Euclidean Distance

1. Buat Fungsi Euclidean Distance untuk menghitung jarak dari data ke centroid dan juga berguna untuk menghitung jarak antara centroid

In [4]:
```
def euclide(x1,x2,ax=1):
    return np.linalg.norm(x1-x2,axis=ax)
```

D.2 Initial Centroid

1. Tentukan terlebih dahulu nilai k sebagai banyaknya cluster yang akan dihitung nilai SSE dari cluster tersebut
2. Tentukan nilai random untuk nilai centroid pada sumbu x dan sumbu y.
3. Titik centroid_x dan centroid_y digabungkan menjadi satu array sehingga diperoleh nilai centroid yang merupakan **initial centroid**

In [5]:
```
k = 4
centroid_x = np.random.randint(0, np.max(data)-5, size=k)
centroid_y = np.random.randint(0, np.max(data)-5, size=k)
centroid = np.array(list(zip(centroid_x, centroid_y)),dtype=np.float32)
```

D.3 Update Centroid

1. Buat variabel untuk menampung centroid lama, yaitu centroid yang belum di update. Centroid lama tersebut akan digunakan untuk menghitung jarak antara centroid lama dengan centroid baru(yang sudah diupdate)
2. Buat variabel untuk menampung jenis cluster dari tiap data, sehingga data dengan indeks ke i setelah proses perhitungan jarak akan memiliki nilai clusternya
3. Buat variabel untuk menampung jarak antara centroid lama dengan centroid baru(yang sudah diupdate)

4. Lakukan perulangan selama jarak antara centroid lama dengan centroid baru(yang sudah diupdate) sudah bernilai 0. Artinya letak centroid lama dan centroid baru sudah tidak berpindah pindah lagi.
5. Selama perulangan tersebut, lakukan: 5.1 Hitung jarak antara seluruh data dengan titik titik centroid, setelah itu cari data mana yang jaraknya paling kecil dengan suatu centroid, maka data tersebut masuk kedalam kelompok cluster centroid tersebut. Hal ini dilakukan selama masih dalam range dari panjang data train. 5.2 Centroid yang sudah dihitung jaraknya terhadap data, jadikan sebagai **centroid lama**. 5.3 Selama panjang cluster, cocokkan nilai cluster antara tiap data dan masukkan kedalam suatu variable baru yang menampung data data di cluster yang sama. Setelah itu, akan dihitung **rata-rata** dari tiap data di cluster yang sama. **Rata-rata** tersebut akan menjadi nilai **centroid baru**
6. Tampilkan nilai centroid baru yang sudah tidak berpindah pindah lagi

In [6]:
```
centroid_old = np.zeros(centroid.shape)
cluster = np.zeros(len(data))
error = euclide(centroid,centroid_old,None)
while error.all() != 0:
    for i in range(len(data)):
        distances = euclide(data[i], centroid)
        c = np.argmin(distances)
        cluster[i] = c
    centroid_old = deepcopy(centroid)
    for i in range(k):
        points = [data[j] for j in range(len(data)) if cluster[j] == i]
        centroid[i] = np.mean(points, axis=0)
    error = euclide(centroid, centroid_old)
print("Centroid: ", centroid)
```

```
Centroid:  [[32.158928  17.869642 ]
 [21.630474   6.7822485]
 [11.0149355  7.5305195]
 [10.685799  22.762426 ]]
```

D.4 Pengelompokan Data Pada Tiap Cluster

1. Buat variable penampung yang akan menampung data pada tiap cluster yang berbeda. Variable tersebut sejumlah dengan jumlah cluster yang digunakan
2. Lakukan pengelompokan selama panjang data: 2.1 Jika nilai cluster pada data yang sudah dihitung di bagian **D.3 Update Centroid** bernilai 0.0 maka masukkan indeks dari data tersebut ke cluster0. 2.2 Jika nilai cluster pada data bernilai 1.0 maka masukkan indeks dari data tersebut ke cluster1, dst.
3. Tampilkan indeks dari tiap data yang ada pada suatu cluster dengan banyaknya data pada cluster tersebut.

In [7]:
```
cluster0 = []
cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []
cluster5 = []
cluster6 = []
cluster7 = []
cluster8 = []
for i in range(len(data)):
    if cluster[i]==0.0:
        cluster0.append(i)
    elif cluster[i]==1.0:
        cluster1.append(i)
    elif cluster[i]==2.0:
        cluster2.append(i)
    elif cluster[i]==3.0:
        cluster3.append(i)
    elif cluster[i]==4.0:
        cluster4.append(i)
    elif cluster[i]==5.0:
        cluster5.append(i)
    elif cluster[i]==6.0:
        cluster6.append(i)
    elif cluster[i]==7.0:
        cluster7.append(i)
    elif cluster[i]==8.0:
        cluster8.append(i)
print("Cluster 0: ",cluster0, "Jumlah data: ",len(cluster0))
```

```
print()
print("Cluster 1: ",cluster1, "Jumlah data: ",len(cluster1))
print()
print("Cluster 2: ",cluster2, "Jumlah data: ",len(cluster2))
print()
print("Cluster 3: ",cluster3, "Jumlah data: ",len(cluster3))
print()
print("Cluster 4: ",cluster4, "Jumlah data: ",len(cluster4))
print()
print("Cluster 5: ",cluster5, "Jumlah data: ",len(cluster5))
print()
print("Cluster 6: ",cluster6, "Jumlah data: ",len(cluster6))
print()
print("Cluster 7: ",cluster7, "Jumlah data: ",len(cluster7))
print()
print("Cluster 8: ",cluster8, "Jumlah data: ",len(cluster8))
```

```
Cluster 0:  [22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 3
6, 37, 38, 206, 207, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219,
220, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 2
36, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 25
0, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 26
4, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 547, 54
8, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 56
2, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 57
6, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 59
0, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 60
4, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 61
8, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 63
2, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 64
6, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658] Jumlah d
ata:  196

Cluster 1:  [186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 19
7, 198, 199, 200, 201, 202, 203, 204, 205, 208, 209, 221, 222, 289, 29
2, 364, 365, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 38
3, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 39
7, 398, 399, 400, 401, 402, 403, 404, 407, 408, 409, 410, 411, 412, 41
3, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 42
7, 428, 429, 430, 431, 434, 435, 436, 437, 438, 439, 440, 441, 442, 44
```

```
3, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 45
7, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 47
1, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 48
5, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 49
9, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 51
3, 514, 515, 516] Jumlah data:  169

Cluster 2:  [277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 28
8, 290, 291, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 30
4, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 31
8, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 33
2, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 34
6, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 36
0, 361, 362, 363, 366, 367, 368, 369, 370, 371, 405, 406, 432, 433, 51
7, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 53
1, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 54
5, 546, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 67
1, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 68
5, 686, 687] Jumlah data:  154

Cluster 3:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
17, 18, 19, 20, 21, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69,
70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87,
88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 10
4, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 11
8, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 13
2, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 14
6, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 16
0, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 17
4, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185] Jumlah data:
169

Cluster 4:  [] Jumlah data:  0

Cluster 5:  [] Jumlah data:  0

Cluster 6:  [] Jumlah data:  0
```

```
Cluster 7:  [] Jumlah data:  0

Cluster 8:  [] Jumlah data:  0
```

E. Perhitungan Quality of Cluster menggunakan SSE

Kualitas bagus atau tidaknya cluster yang kita pilih dengan nilai centroid yang diperoleh, dapat diukur menggunakan perhitungan Sum of Square Error (SSE). Berikut adalah rumusnya:
Cara penggunaan rumus ini adalah:

1. Lakukan perulangan selama panjang dari cluster
2. Selama perulangan, lakukan: Hitung jarak dari tiap data dengan centroid selama panjang data dari tiap cluster, dengan aturan tiap data berada pada cluster yang sama
3. Setelah memperoleh total jarak antara data dengan centroid dari tiap cluster, maka jumlahkan seluruh data tersebut sehingga diperoleh nilai **SSE** yang diinginkan.

In [8]:
```
total0 = 0
total1 = 0
total2 = 0
total3 = 0
total4 = 0
total5 = 0
total6 = 0
total7 = 0
total8 = 0
for i in range(k):
    total0 = np.sum([euclide(data[j],centroid[i], ax=0) for j in range(
len(cluster0)) if cluster[j] == i])
    total1 = np.sum([euclide(data[j],centroid[i], ax=0) for j in range(
len(cluster1)) if cluster[j] == i])
    total2 = np.sum([euclide(data[j],centroid[i], ax=0) for j in range(
len(cluster2)) if cluster[j] == i])
    total3 = np.sum([euclide(data[j],centroid[i], ax=0) for j in range(
len(cluster3)) if cluster[j] == i])
    total4 = np.sum([euclide(data[j],centroid[i], ax=0) for j in range(
len(cluster4)) if cluster[j] == i])
```

```
    total5 = np.sum([euclide(data[j],centroid[i], ax=0) for j in range(
len(cluster5)) if cluster[j] == i])
    total6 = np.sum([euclide(data[j],centroid[i], ax=0) for j in range(
len(cluster6)) if cluster[j] == i])
    total7 = np.sum([euclide(data[j],centroid[i], ax=0) for j in range(
len(cluster7)) if cluster[j] == i])
    total8 = np.sum([euclide(data[j],centroid[i], ax=0) for j in range(
len(cluster8)) if cluster[j] == i])
print("Total Jarak Cluster 0: ",total0)
print("Total Jarak Cluster 1: ",total1)
print("Total Jarak Cluster 2: ",total2)
print("Total Jarak Cluster 3: ",total3)
print("Total Jarak Cluster 4: ",total4)
print("Total Jarak Cluster 5: ",total5)
print("Total Jarak Cluster 6: ",total6)
print("Total Jarak Cluster 7: ",total7)
print("Total Jarak Cluster 8: ",total8)
print("SSE: ",total0+total1+total2+total3+total4+total5+total6+total7+t
otal8)
```

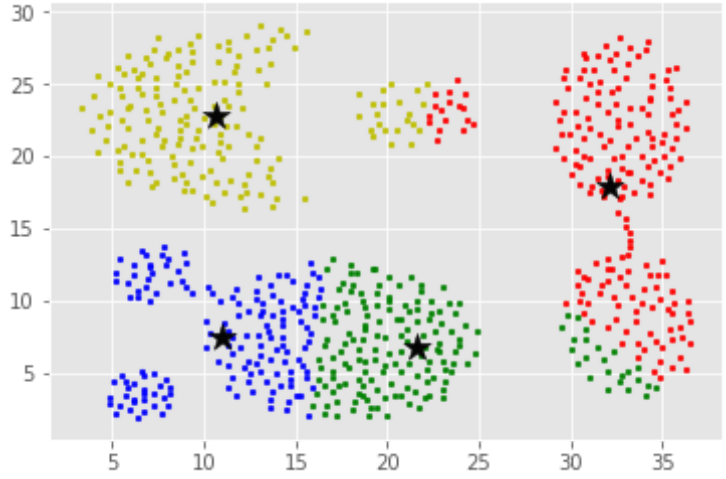
```
Total Jarak Cluster 0:  851.6964773717432
Total Jarak Cluster 1:  760.6058741819038
Total Jarak Cluster 2:  723.4802625071152
Total Jarak Cluster 3:  760.6058741819038
Total Jarak Cluster 4:  0.0
Total Jarak Cluster 5:  0.0
Total Jarak Cluster 6:  0.0
Total Jarak Cluster 7:  0.0
Total Jarak Cluster 8:  0.0
SSE:  3096.388488242666
```

F. Visualisasi Data yang Telah di Clustering

In [9]:
```
colors = ['r', 'g', 'b', 'y', 'c', 'm','black','grey','orange','purple'
]
fig, ax = plt.subplots()
for i in range(k):
    points = np.array([data[j] for j in range(len(data)) if cluster[j]
```

```
== i])
    ax.scatter(points[:, 0], points[:, 1], s=7, c=colors[i])
ax.scatter(centroid[:, 0], centroid[:, 1], marker='*', s=200, c='#05050
5')
```

Out[9]: <matplotlib.collections.PathCollection at 0xf0d8650>



G. Observasi Data Train

Elbow Method

Observasi data train yang saya lakukan adalah observasi tiap nilai **SSE** dari suatu nilai cluster yang ditentukan dengan nilai centroid yang sudah fix. Untuk melakukan hal tersebut, saya mencoba untuk melakukan perhitungan nilai **SSE** pada nilai cluster dengan **Range(1,8)** .
Pada program ini, hal tersebut dapat dilakukan dengan cara:

1. Mengganti nilai **k** yang ada pada **code** seperti pada point ke **--D.2 Initial Centroid--**. Setelah mengganti nilai **k** dan memperoleh SSE dan Centroid, maka:
2. Nilai SSE dan nilai centroid dari SSE tersebut disimpan kedalam suatu file excel.

3. Nilai SSE dari tiap cluster dapat divisualisasikan dan dapat dimodelkan menjadi suatu grafik
4. Dari grafik tersebut, dapat ditentukan nilai cluster mana yang dapat diperoleh sesuai dengan **Elbow Method.**

   Berikut adalah data dari nilai SSE dan Centroid dari tiap cluster yang saya observasi:

In [10]:
```
raw_data1 = {'k': [1],
             'Centroid x': [19.565262],
             'Centroid y': [14.033721],
             'SSE' :[8322.673788]}
c1 = pd.DataFrame(raw_data1,columns = ['k', 'Centroid x','Centroid y',
'SSE'])
c1
```

Out[10]:

	k	Centroid x	Centroid y	SSE
0	1	19.565262	14.033721	8322.673788

In [11]:
```
raw_data2 = {'k': [2,''],
             'Centroid x': [31.582619,13.411319],
             'Centroid y': [16.663734,12.686923],
             'SSE' :[4931.087545,'']}
c2 = pd.DataFrame(raw_data2,columns = ['k','Centroid x','Centroid y',
'SSE'])
c2
```

Out[11]:

	k	Centroid x	Centroid y	SSE
0	2	31.582619	16.663734	4931.09
1		13.411319	12.686923	

In [12]:
```
raw_data3 = {'k': [3,'',''],
             'Centroid x': [32.30764,15.485204,10.841573],
```

```
            'Centroid y': [16.432638, 7.1409864, 22.507303],
            'SSE' :[2662.359933,'',''],}
c3 = pd.DataFrame(raw_data3,columns = ['k','Centroid x','Centroid y',
'SSE'])
c3
```

Out[12]:

	k	Centroid x	Centroid y	SSE
0	3	32.307640	16.432638	2662.36
1		15.485204	7.140986	
2		10.841573	22.507303	

In [13]:

```
raw_data4 = {'k': [4,'','',''],
            'Centroid x': [17.767733,
                           12.230082,
                           10.859322,
                           32.30764,],
            'Centroid y': [8.754651,
                           4.9341464,
                           22.559605,
                           16.432638,],
            'SSE' :[901.3766958,'','',''],}
c4 = pd.DataFrame(raw_data4,columns = ['k','Centroid x','Centroid y',
'SSE'])
c4
```

Out[13]:

	k	Centroid x	Centroid y	SSE
0	4	17.767733	8.754651	901.377
1		12.230082	4.934146	
2		10.859322	22.559605	
3		32.307640	16.432638	

In [14]:
```
raw_data5 = {'k': [5,'','','',''],
             'Centroid x': [18.100231,
                            32.73835,
                            13.921818,
                            8.47623,
                            7.9863157,],
             'Centroid y': [7.1824074,
                            16.09733,
                            24.465,
                            5.637705,
                            18.449474,],
             'SSE' :[652.8890597,'','','',''']}
c5 = pd.DataFrame(raw_data5,columns = ['k','Centroid x','Centroid y',
'SSE'])
c5
```

Out[14]:

	k	Centroid x	Centroid y	SSE
0	5	18.100231	7.182407	652.889
1		32.738350	16.097330	
2		13.921818	24.465000	
3		8.476230	5.637705	
4		7.986316	18.449474	

In [15]:
```
raw_data6 = {'k': [6,'','','','',''],
             'Centroid x': [10.295193,
                            8.575,
                            26.455084,
                            32.77621,
                            12.546482,
                            21.47,],
             'Centroid y': [26.3375,
                                20.648958,
                                7.4409604,
                                21.072178,
```

```
                                    7.3271356,
                                    22.88125,],
              'SSE' :[479.6905512,'','','','','']}
c6 = pd.DataFrame(raw_data6,columns = ['k','Centroid x','Centroid y',
'SSE'])
c6
```

Out[15]:

	k	Centroid x	Centroid y	SSE
0	6	10.295193	26.337500	479.691
1		8.575000	20.648958	
2		26.455084	7.440960	
3		32.776210	21.072178	
4		12.546482	7.327136	
5		21.470000	22.881250	

In [16]:

```
raw_data7 = {'k': [7,'','','','','','',''],
             'Centroid x': [11.914286,
                            14.806048,
                            30.825834,
                            32.655857,
                            8.612,
                            19.34091,
                            23.135],
             'Centroid y': [14.994047,
                            6.7435484,
                            8.377084,
                            22.051802,
                            23.3392,
                            23.461363,
                            23.325,],
             'SSE' :[138.1096539,'','','','','','',''']}
c7 = pd.DataFrame(raw_data7,columns = ['k','Centroid x','Centroid y',
```

```
'SSE'])
c7
```

Out[16]:

	k	Centroid x	Centroid y	SSE
0	7	11.914286	14.994047	138.11
1		14.806048	6.743548	
2		30.825834	8.377084	
3		32.655857	22.051802	
4		8.612000	23.339200	
5		19.340910	23.461363	
6		23.135000	23.325000	

In [17]:

```
raw_data8 = {'k': [8,'','','','','','',''],
             'Centroid x': [16.07037,
                            8.881372,
                            10.109821,
                            18.823404,
                            7.082258,
                            31.615854,
                            33.10798,
                            9.92],
             'Centroid y': [20.82037,
                            4.1284313,
                            26.202679,
                            7.0018616,
                            21.314516,
                            22.306911,
                            8.855319,
                            10.65
                            ],
             'SSE' :[0,'','','','','','',''']}
c8 = pd.DataFrame(raw_data8,columns = ['k','Centroid x','Centroid y',
```

```
'SSE'])
c8
```

Out[17]:

	k	Centroid x	Centroid y	SSE
0	8	16.070370	20.820370	0
1		8.881372	4.128431	
2		10.109821	26.202679	
3		18.823404	7.001862	
4		7.082258	21.314516	
5		31.615854	22.306911	
6		33.107980	8.855319	
7		9.920000	10.650000	

Dari nilai **SSE** pada setiap cluster tersebut, maka dapat memperoleh grafik sebagai berikut:

Sehingga, pada grafik tersebut dapat dilihat bahwa sesuai dengan **Elbow Method** saya memilih **nilai cluster** yaitu **4** dengan **nilai SSE** pada data train yaitu sebesar **901.3766958**.

PENGUJIAN DATA TEST

Pada data test, akan dilakukan pengujian terhadap nilai cluster yang telah diperoleh pada tahap observasi data train yaitu **Nilai Cluster = 4** dengan **nilai centroid x = [17.767733, 12.230082, 10.859322, 32.30764]** dan **nilai centroid y = [8.754651, 4.9341464, 22.559605, 16.432638]** sehingga kita akan memperoleh tipe cluster dari tiap data dan nilai SSE pada data test tersebut.

A. Load Data Test

In [18]:
```
filetest = pd.read_csv('Data Test.csv')
filetest
```

Out[18]:

	Value1	Value2
0	18.75	22.95
1	21.45	21.45
2	20.50	22.85
3	20.65	24.30
4	21.70	23.80
5	23.10	21.70
6	13.35	28.45
7	12.40	27.85
8	12.20	28.65
9	12.90	26.50
10	11.15	28.70
11	10.50	28.35
12	10.25	27.25
13	12.60	24.05
14	10.05	25.95
15	8.50	27.05
16	7.55	26.30
17	9.40	25.55

	Value1	Value2
18	10.55	24.35
19	5.40	25.25
20	4.30	24.00
21	6.10	22.60
22	6.40	21.95
23	8.45	17.20
24	12.30	22.75
25	9.95	19.80
26	12.00	20.00
27	11.40	19.25
28	15.20	18.20
29	31.90	4.40
...	...	...
70	21.75	8.20
71	23.00	7.35
72	23.70	8.85
73	5.15	3.45
74	4.95	4.05
75	7.10	4.30
76	8.50	3.25
77	32.45	16.75
78	30.55	18.80

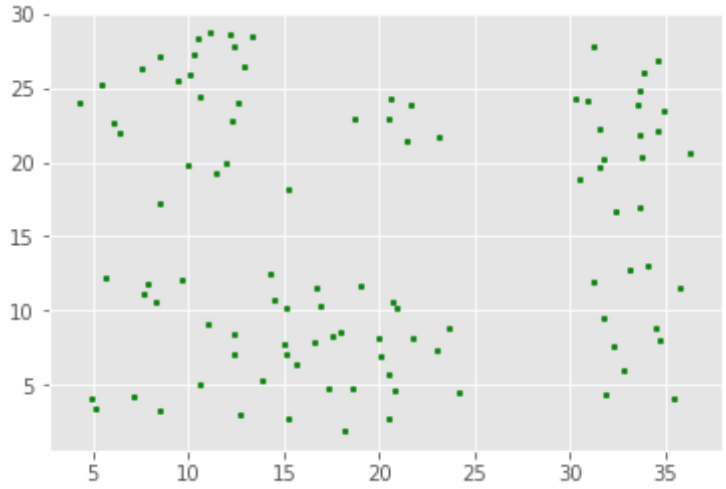
	Value1	Value2
79	31.55	19.65
80	33.70	17.00
81	31.75	20.25
82	31.55	22.20
83	30.95	24.15
84	33.65	21.90
85	33.80	20.40
86	36.35	20.60
87	34.60	22.05
88	34.90	23.50
89	33.60	23.90
90	33.70	24.85
91	30.25	24.30
92	31.25	27.85
93	33.85	26.05
94	34.65	26.85
95	9.70	12.10
96	5.70	12.25
97	7.85	11.85
98	7.65	11.10
99	8.30	10.55

100 rows × 2 columns

B. Visualisasi Data

In [25]:
```
value1 = filetest['Value1'].values
value2 = filetest['Value2'].values
data = np.array(list(zip(value1, value2)))
plt.scatter(value1, value2, c='green', s=7)
```

Out[25]: <matplotlib.collections.PathCollection at 0x3bbb90>



C. Clustering Data Test

C.1 Nilai Cluster dan Centroid

In [26]:
```
k = 4
centroid_x = np.array([17.767733, 12.230082, 10.859322, 32.30764])
centroid_y = np.array([8.754651, 4.9341464, 22.559605,16.432638])
centroid = np.array(list(zip(centroid_x, centroid_y)),dtype=np.float32)
```

C.2 Perhitungan Jarak Data dengan Centroid

In [27]:
```
cluster = np.zeros(len(data))
for i in range(len(data)):
    distances = euclide(data[i], centroid)
    c = np.argmin(distances)
    cluster[i] = c
```

C.3 Pengelompokan Tiap Data

In [28]:
```
cluster0 = []
cluster1 = []
cluster2 = []
cluster3 = []
clustering = []
for i in range(len(data)):
    if cluster[i]==0.0:
        cluster0.append(i)
    elif cluster[i]==1.0:
        cluster1.append(i)
    elif cluster[i]==2.0:
        cluster2.append(i)
    elif cluster[i]==3.0:
        cluster3.append(i)
print("Cluster 0: ",cluster0, "Jumlah data: ",len(cluster0))
print()
print("Cluster 1: ",cluster1, "Jumlah data: ",len(cluster1))
print()
print("Cluster 2: ",cluster2, "Jumlah data: ",len(cluster2))
print()
print("Cluster 3: ",cluster3, "Jumlah data: ",len(cluster3))
print()
```

```
Cluster 0:  [41, 42, 45, 46, 47, 51, 53, 54, 56, 57, 58, 59, 60, 61, 6
2, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72] Jumlah data:  25

Cluster 1:  [40, 43, 44, 48, 49, 50, 52, 55, 73, 74, 75, 76, 95, 96, 9
```

```
7, 98, 99] Jumlah data:  17

Cluster 2:  [0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28] Jumlah data:  28

Cluster 3:  [5, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 77, 78, 79,
80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94] Jumlah dat
a:  30
```

C.4 Membuat File Prediksi

In [29]:
```
raw_data = [{'Values1': [file.loc[i,'Value1']],
             'Values2': [file.loc[i,'Value2']],
             'Cluster': cluster[i]} for i in range(len(data))]
df = pd.DataFrame(raw_data,columns = ['Values1', 'Values2','Cluster'])
df.to_csv("File_Clustering.csv")
df
```

Out[29]:

	Values1	Values2	Cluster
0	[21.3]	[20.8]	2.0
1	[20.15]	[20.9]	2.0
2	[19.2]	[21.35]	2.0
3	[19.1]	[21.85]	2.0
4	[18.45]	[22.8]	2.0
5	[19.4]	[23.0]	3.0
6	[19.55]	[22.25]	2.0
7	[19.8]	[21.85]	2.0
8	[20.5]	[21.85]	2.0

	Values1	Values2	Cluster
9	[21.7]	[21.9]	2.0
10	[21.4]	[22.3]	2.0
11	[21.0]	[22.6]	2.0
12	[21.15]	[22.95]	2.0
13	[19.75]	[23.65]	2.0
14	[19.2]	[23.7]	2.0
15	[18.45]	[24.35]	2.0
16	[20.65]	[23.85]	2.0
17	[19.7]	[24.6]	2.0
18	[20.15]	[25.05]	2.0
19	[22.15]	[25.1]	2.0
20	[21.6]	[24.65]	2.0
21	[21.9]	[23.65]	2.0
22	[22.55]	[23.5]	2.0
23	[22.55]	[24.3]	2.0
24	[23.3]	[24.45]	2.0
25	[24.25]	[24.35]	2.0
26	[23.8]	[25.25]	2.0
27	[23.4]	[23.8]	2.0
28	[22.9]	[23.2]	2.0
29	[22.3]	[22.8]	3.0
...	...	...	...

	Values1	Values2	Cluster
70	[7.3]	[27.4]	0.0
71	[6.8]	[26.85]	0.0
72	[7.0]	[26.5]	0.0
73	[8.55]	[26.3]	1.0
74	[9.0]	[25.85]	1.0
75	[8.6]	[25.65]	1.0
76	[8.45]	[25.05]	1.0
77	[8.85]	[24.6]	3.0
78	[9.65]	[24.7]	3.0
79	[11.05]	[23.9]	3.0
80	[10.55]	[23.55]	3.0
81	[9.45]	[23.35]	3.0
82	[9.2]	[23.9]	3.0
83	[8.35]	[23.9]	3.0
84	[7.35]	[24.75]	3.0
85	[7.4]	[25.45]	3.0
86	[6.6]	[25.75]	3.0
87	[6.1]	[26.0]	3.0
88	[5.8]	[26.95]	3.0
89	[5.65]	[25.8]	3.0
90	[5.3]	[26.1]	3.0
91	[6.4]	[25.4]	3.0

	Values1	Values2	Cluster
92	[5.35]	[24.7]	3.0
93	[4.8]	[25.05]	3.0
94	[4.2]	[25.55]	3.0
95	[6.4]	[24.8]	1.0
96	[6.55]	[24.3]	1.0
97	[7.4]	[24.25]	1.0
98	[5.45]	[24.2]	1.0
99	[4.0]	[24.25]	1.0

100 rows × 3 columns

D. Perhitungan Quality of Cluster menggunakan SSE

D.1 SSE

In [30]:
```
total0 = 0
total1 = 0
total2 = 0
total3 = 0
total4 = 0
for i in range(k):
    total0 = np.sum([euclide(data[j],centroid[i], ax=0) for j in range(
len(cluster0)) if cluster[j] == i])
    total1 = np.sum([euclide(data[j],centroid[i], ax=0) for j in range(
len(cluster1)) if cluster[j] == i])
    total2 = np.sum([euclide(data[j],centroid[i], ax=0) for j in range(
len(cluster2)) if cluster[j] == i])
    total3 = np.sum([euclide(data[j],centroid[i], ax=0) for j in range(
len(cluster3)) if cluster[j] == i])
```

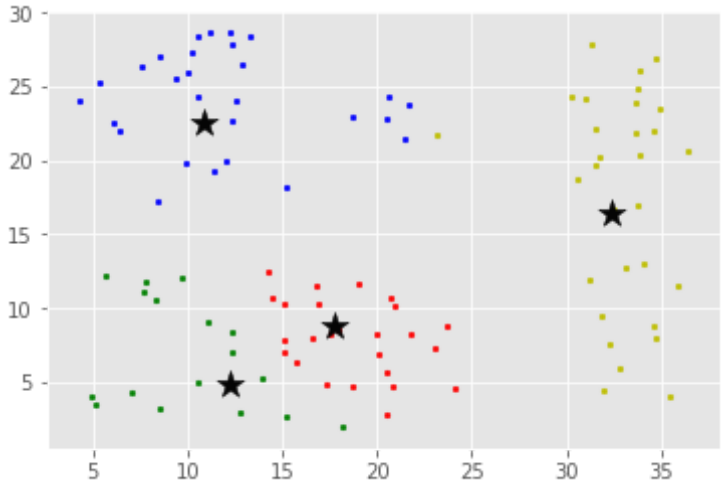
```
print("Total Jarak Cluster 0: ",total0)
print("Total Jarak Cluster 1: ",total1)
print("Total Jarak Cluster 2: ",total2)
print("Total Jarak Cluster 3: ",total3)
print("SSE: ",total0+total1+total2+total3)
```

```
Total Jarak Cluster 0:  10.607814875317922
Total Jarak Cluster 1:  10.607814875317922
Total Jarak Cluster 2:  10.607814875317922
Total Jarak Cluster 3:  22.647356051017876
SSE:  54.47080067697165
```

D.2 Visualisasi Clustering

In [31]:
```
colors = ['r', 'g', 'b', 'y', 'c', 'm','black','grey','orange','purple'
]
fig, ax = plt.subplots()
for i in range(k):
    points = np.array([data[j] for j in range(len(data)) if cluster[j]
== i])
    ax.scatter(points[:, 0], points[:, 1], s=7, c=colors[i])
ax.scatter(centroid[:, 0], centroid[:, 1], marker='*', s=200, c='#05050
5')
```

Out[31]:
```
<matplotlib.collections.PathCollection at 0x2a75d90>
```



KESIMPULAN

Kesimpulan dari pengujian data test menggunakan Nilai Cluster = 4 dengan nilai centroid x = [17.767733, 12.230082, 10.859322, 32.30764] dan nilai centroid y = [8.754651, 4.9341464, 22.559605, 16.432638] adalah:

SSE = 54.47080067697165

Sehingga, dapat dikatakan bahwa hasil observasi data train yang digunakan pada data test, membuat kualitas dari nilai cluster dengan centroid tersebut menjadi baik pada data test dengan nilai SSE yang cukup kecil.