Tugas 2: K-Means Clustering

Nama : Septiana Putri
Nim : 1301154462
Kelas : IF-39-14

Menginputkan library yang dibutuhkan

a. Numpy = package untuk scientific computing
b. Pandas = data analis toolkit untuk membaca data dengan jenis apapun
c. Plotly = untuk visualisasi

In [1]:
```
import numpy
import numpy as num
import pandas
import pandas as pd
from matplotlib import pyplot as plt
from copy import deepcopy
```

Meload Data Train

In [2]:
```
dtrain = pd.read_csv("TrainsetTugas2.txt",delimiter="\t")
```

Memisahkan data train menjadi dua value

In [3]:
```
a = dtrain["v1"].values
```

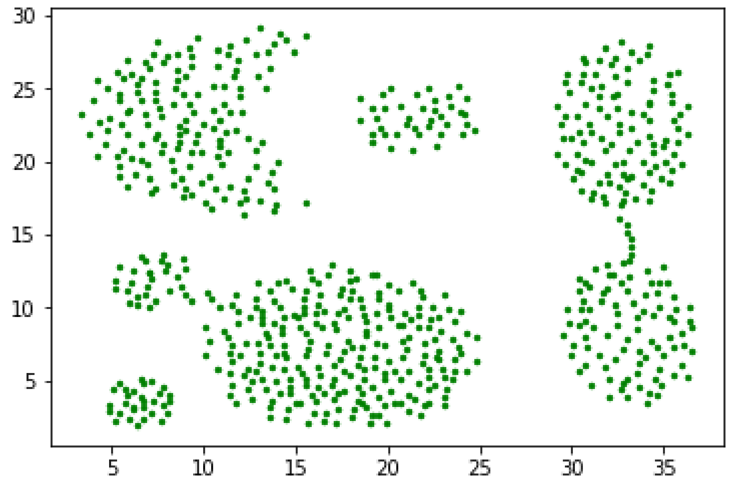
```
b = dtrain["v2"].values
```

Visualisasi Data Train

Melakukan visualisasi data train kedalam scatter dengan keseluruhan menggunakan warna hijau yaitu masih dalam bagian yang sama.

In [4]:
```
dt1 = num.array(list(zip(a,b)))
plt.scatter(a,b,c="green",s=6)
```

Out[4]:
```
<matplotlib.collections.PathCollection at 0xb74ed30>
```



Clustering Data Train

Dengan menggunakan fungsi untuk menghitung Euclidean Distance atau menghitung jarak dari data ke centroid dan menghitung jarak antar centroid

In [5]:
```
def hitdistance(x,y,a=1):
    return num.linalg.norm(x-y,axis=a)
```

Inisiasi Centroid

menentukan nilai clust sebagai banyaknya cluster yang akan dihitung nilai SSE dari cluster tersebut. Kemudian tentukan nilai random untuk nilai centroid pada sumbu x dan sumbu y.didapatkan nilai centroid yang merupakan initial centroid dari px dan py.

In [6]:
```
clust = 5

ps = dt1.copy()
px = num.random.randint(0,num.max(ps)-10,size=clust)
py = num.random.randint(0,num.max(ps)-10,size=clust)
pxy = num.array(list(zip(px,py)))
print(pxy)
```

```
[[21  2]
 [16 17]
 [ 7  0]
 [18 22]
 [21  9]]
```

Update Nilai Centroid

Buat variabel penampung centroid belum di update yang akan digunakan untuk menghitung jarak antara centroid lama dengan centroid yang sudah diupdate. Buat variabel untuk menampung jenis cluster dari tiap data, sehingga data dengan indeks ke i setelah proses perhitungan jarak akan memiliki nilai clusternya dan kemudian di tampung di suatu variabel. lakukan perulangan hingga centroid lama dan baru tidak berubah lagi dan menampilkan nilai centroid baru tersebut

In [7]:
```
s_old  = 0
k = num.zeros(len(dt1))

while True:
    for i in range (len(dt1)):
        dnew = hitdistance(dt1[i],pxy)
        win = num.argmin(dnew)
        k[i]= win
    s_old = deepcopy(pxy)
    for i in range (clust):
        node = [dt1[j]for j in range (len(dt1)) if k[j]==i]
        pxy[i] = num.mean(node,axis=0)
    newjarak = hitdistance(pxy,s_old)
    if newjarak.all() == 0:
        break
print(pxy)
```

```
[[18  7]
 [ 9 22]
 [ 9  7]
 [29 22]
 [32  8]]
```

In [8]:
```
for i in range (len(dt1)):
    print(i,k[i])
```

```
0 3.0
1 3.0
2 3.0
3 3.0
4 1.0
5 3.0
6 3.0
7 3.0
8 3.0
9 3.0
10 3.0
11 3.0
12 3.0
13 3.0
14 3.0
15 1.0
16 3.0
17 3.0
```

```
18 3.0
19 3.0
20 3.0
21 3.0
22 3.0
23 3.0
24 3.0
25 3.0
26 3.0
27 3.0
28 3.0
29 3.0
30 3.0
31 3.0
32 3.0
33 3.0
34 3.0
35 3.0
36 3.0
37 3.0
38 3.0
39 1.0
40 1.0
41 1.0
42 1.0
43 1.0
44 1.0
45 1.0
46 1.0
47 1.0
48 1.0
49 1.0
50 1.0
51 1.0
52 1.0
53 1.0
54 1.0
55 1.0
56 1.0
```

```
57 1.0
58 1.0
59 1.0
60 1.0
61 1.0
62 1.0
63 1.0
64 1.0
65 1.0
66 1.0
67 1.0
68 1.0
69 1.0
70 1.0
71 1.0
72 1.0
73 1.0
74 1.0
75 1.0
76 1.0
77 1.0
78 1.0
79 1.0
80 1.0
81 1.0
82 1.0
83 1.0
84 1.0
85 1.0
86 1.0
87 1.0
88 1.0
89 1.0
90 1.0
91 1.0
92 1.0
93 1.0
94 1.0
95 1.0
```

```
96 1.0
97 1.0
98 1.0
99 1.0
100 1.0
101 1.0
102 1.0
103 1.0
104 1.0
105 1.0
106 1.0
107 1.0
108 1.0
109 1.0
110 1.0
111 1.0
112 1.0
113 1.0
114 1.0
115 1.0
116 1.0
117 1.0
118 1.0
119 1.0
120 1.0
121 1.0
122 1.0
123 1.0
124 1.0
125 1.0
126 1.0
127 1.0
128 1.0
129 1.0
130 1.0
131 1.0
132 1.0
133 1.0
134 1.0
```

```
135 1.0
136 1.0
137 1.0
138 1.0
139 1.0
140 1.0
141 1.0
142 1.0
143 1.0
144 1.0
145 1.0
146 1.0
147 1.0
148 1.0
149 1.0
150 1.0
151 1.0
152 1.0
153 1.0
154 1.0
155 1.0
156 1.0
157 1.0
158 1.0
159 1.0
160 1.0
161 1.0
162 1.0
163 1.0
164 1.0
165 1.0
166 1.0
167 1.0
168 1.0
169 1.0
170 1.0
171 1.0
172 1.0
173 1.0
```

```
174 1.0
175 1.0
176 1.0
177 1.0
178 1.0
179 1.0
180 1.0
181 1.0
182 1.0
183 1.0
184 1.0
185 1.0
186 4.0
187 4.0
188 4.0
189 4.0
190 4.0
191 4.0
192 4.0
193 4.0
194 4.0
195 4.0
196 4.0
197 4.0
198 4.0
199 4.0
200 4.0
201 4.0
202 4.0
203 4.0
204 4.0
205 4.0
206 4.0
207 4.0
208 4.0
209 4.0
210 4.0
211 4.0
212 4.0
```

```
213 4.0
214 4.0
215 4.0
216 4.0
217 4.0
218 4.0
219 4.0
220 4.0
221 4.0
222 4.0
223 4.0
224 4.0
225 4.0
226 4.0
227 4.0
228 4.0
229 4.0
230 4.0
231 4.0
232 4.0
233 4.0
234 4.0
235 4.0
236 4.0
237 4.0
238 4.0
239 4.0
240 4.0
241 4.0
242 4.0
243 4.0
244 4.0
245 4.0
246 4.0
247 4.0
248 4.0
249 4.0
250 4.0
251 4.0
```

```
252 4.0
253 4.0
254 4.0
255 4.0
256 4.0
257 4.0
258 4.0
259 4.0
260 4.0
261 4.0
262 4.0
263 4.0
264 4.0
265 4.0
266 4.0
267 4.0
268 4.0
269 4.0
270 4.0
271 4.0
272 4.0
273 4.0
274 4.0
275 4.0
276 4.0
277 2.0
278 2.0
279 2.0
280 2.0
281 2.0
282 2.0
283 2.0
284 2.0
285 2.0
286 0.0
287 0.0
288 0.0
289 0.0
290 0.0
```

```
291 0.0
292 0.0
293 0.0
294 0.0
295 0.0
296 0.0
297 2.0
298 2.0
299 2.0
300 2.0
301 2.0
302 2.0
303 2.0
304 2.0
305 2.0
306 2.0
307 0.0
308 2.0
309 2.0
310 2.0
311 2.0
312 2.0
313 2.0
314 2.0
315 0.0
316 0.0
317 0.0
318 0.0
319 0.0
320 0.0
321 0.0
322 0.0
323 0.0
324 0.0
325 0.0
326 0.0
327 0.0
328 0.0
329 0.0
```

```
330 0.0
331 0.0
332 2.0
333 2.0
334 2.0
335 2.0
336 2.0
337 2.0
338 2.0
339 2.0
340 0.0
341 2.0
342 2.0
343 2.0
344 2.0
345 2.0
346 2.0
347 2.0
348 2.0
349 0.0
350 0.0
351 0.0
352 0.0
353 0.0
354 0.0
355 0.0
356 0.0
357 0.0
358 0.0
359 0.0
360 0.0
361 0.0
362 0.0
363 0.0
364 0.0
365 0.0
366 0.0
367 0.0
368 0.0
```

```
369 0.0
370 0.0
371 0.0
372 0.0
373 0.0
374 0.0
375 0.0
376 0.0
377 0.0
378 0.0
379 0.0
380 0.0
381 0.0
382 0.0
383 0.0
384 0.0
385 0.0
386 0.0
387 0.0
388 0.0
389 0.0
390 0.0
391 0.0
392 0.0
393 0.0
394 0.0
395 0.0
396 0.0
397 0.0
398 0.0
399 0.0
400 0.0
401 0.0
402 0.0
403 0.0
404 0.0
405 0.0
406 0.0
407 0.0
```

```
408 0.0
409 0.0
410 0.0
411 0.0
412 0.0
413 0.0
414 0.0
415 0.0
416 0.0
417 0.0
418 0.0
419 0.0
420 0.0
421 0.0
422 0.0
423 0.0
424 0.0
425 0.0
426 0.0
427 0.0
428 0.0
429 0.0
430 0.0
431 0.0
432 0.0
433 0.0
434 0.0
435 0.0
436 0.0
437 0.0
438 0.0
439 0.0
440 0.0
441 0.0
442 0.0
443 0.0
444 0.0
445 0.0
446 0.0
```

```
447 0.0
448 0.0
449 0.0
450 0.0
451 0.0
452 0.0
453 0.0
454 0.0
455 0.0
456 0.0
457 0.0
458 0.0
459 0.0
460 0.0
461 0.0
462 0.0
463 0.0
464 0.0
465 0.0
466 0.0
467 0.0
468 0.0
469 0.0
470 0.0
471 0.0
472 0.0
473 0.0
474 0.0
475 0.0
476 0.0
477 0.0
478 0.0
479 0.0
480 0.0
481 0.0
482 0.0
483 0.0
484 0.0
485 0.0
```

```
486 0.0
487 0.0
488 0.0
489 0.0
490 0.0
491 0.0
492 0.0
493 0.0
494 0.0
495 0.0
496 0.0
497 0.0
498 0.0
499 0.0
500 0.0
501 0.0
502 0.0
503 0.0
504 0.0
505 0.0
506 0.0
507 0.0
508 0.0
509 4.0
510 0.0
511 0.0
512 0.0
513 0.0
514 0.0
515 0.0
516 4.0
517 2.0
518 2.0
519 2.0
520 2.0
521 2.0
522 2.0
523 2.0
524 2.0
```

```
525 2.0
526 2.0
527 2.0
528 2.0
529 2.0
530 2.0
531 2.0
532 2.0
533 2.0
534 2.0
535 2.0
536 2.0
537 2.0
538 2.0
539 2.0
540 2.0
541 2.0
542 2.0
543 2.0
544 2.0
545 2.0
546 2.0
547 4.0
548 4.0
549 3.0
550 3.0
551 3.0
552 3.0
553 3.0
554 3.0
555 3.0
556 3.0
557 3.0
558 3.0
559 3.0
560 3.0
561 3.0
562 3.0
563 3.0
```

```
564 3.0
565 3.0
566 3.0
567 3.0
568 3.0
569 3.0
570 3.0
571 3.0
572 3.0
573 3.0
574 3.0
575 3.0
576 3.0
577 3.0
578 3.0
579 3.0
580 3.0
581 3.0
582 3.0
583 3.0
584 3.0
585 3.0
586 3.0
587 3.0
588 3.0
589 3.0
590 3.0
591 3.0
592 3.0
593 3.0
594 3.0
595 3.0
596 3.0
597 3.0
598 3.0
599 3.0
600 3.0
601 3.0
602 3.0
```

```
603 3.0
604 3.0
605 3.0
606 3.0
607 3.0
608 3.0
609 3.0
610 3.0
611 3.0
612 3.0
613 3.0
614 3.0
615 3.0
616 3.0
617 3.0
618 3.0
619 3.0
620 3.0
621 3.0
622 3.0
623 3.0
624 3.0
625 3.0
626 3.0
627 3.0
628 3.0
629 3.0
630 3.0
631 3.0
632 3.0
633 3.0
634 3.0
635 3.0
636 3.0
637 3.0
638 3.0
639 3.0
640 3.0
641 3.0
```

```
642 3.0
643 3.0
644 3.0
645 3.0
646 3.0
647 3.0
648 3.0
649 3.0
650 3.0
651 3.0
652 3.0
653 3.0
654 3.0
655 3.0
656 3.0
657 3.0
658 3.0
659 1.0
660 2.0
661 2.0
662 2.0
663 2.0
664 2.0
665 2.0
666 2.0
667 2.0
668 2.0
669 2.0
670 2.0
671 2.0
672 2.0
673 2.0
674 2.0
675 2.0
676 2.0
677 2.0
678 2.0
679 2.0
680 2.0
```

```
681 2.0
682 2.0
683 2.0
684 2.0
685 2.0
686 2.0
687 2.0
```

In [9]:
```
cl0 = []
cl1 = []
cl2 = []
cl3 = []
cl4 = []
for i in range (len(dt1)):
    if k[i] == 0.0:
        cl0.append(i)
    elif k[i] == 1.0:
        cl1.append(i)
    elif k[i] == 2.0:
        cl2.append(i)
    elif k[i] == 3.0:
        cl3.append(i)
    elif k[i] == 4.0:
        cl4.append(i)

print("cluster 0")
print(cl0)
print("cluster 1")
print(cl1)
print("cluster 2")
print(cl2)
print("cluster 3")
print(cl3)
print("cluster 4")
print(cl4)
```

```
cluster 0
[286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 307, 315, 316,
317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 3
```

```
31, 340, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 36
1, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 37
5, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 38
9, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 40
3, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 41
7, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 43
1, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 44
5, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 45
9, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 47
3, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 48
7, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 50
1, 502, 503, 504, 505, 506, 507, 508, 510, 511, 512, 513, 514, 515]
cluster 1
[4, 15, 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, 104, 105, 106,
107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 1
21, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 13
5, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 14
9, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 16
3, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 17
7, 178, 179, 180, 181, 182, 183, 184, 185, 659]
cluster 2
[277, 278, 279, 280, 281, 282, 283, 284, 285, 297, 298, 299, 300, 301,
302, 303, 304, 305, 306, 308, 309, 310, 311, 312, 313, 314, 332, 333, 3
34, 335, 336, 337, 338, 339, 341, 342, 343, 344, 345, 346, 347, 348, 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, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 67
2, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 68
6, 687]
cluster 3
[0, 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21,
22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 54
9, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 56
3, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 57
7, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 59
1, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 60
```

```
5, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 61
9, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 63
3, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 64
7, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658]
cluster 4
[186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199,
200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 2
14, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 22
8, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 24
2, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 25
6, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 27
0, 271, 272, 273, 274, 275, 276, 509, 516, 547, 548]
```

Menghitung Quality of Cluster dengan SSE

Nilai Sum of Square Error(SSE) digunakan dalam pengukuran kualitas cluster, berikut adalah perhitungannya : a. lakukakn perhitungan sepanjang cluster dengan looping dengan aturan tiap data pada cluster yang sama b. setelah memperoleh nilai total jarak tersebut kemudia dijumlahkan tiap cluster dan nilai tersebut sebagai nilai SSE nya

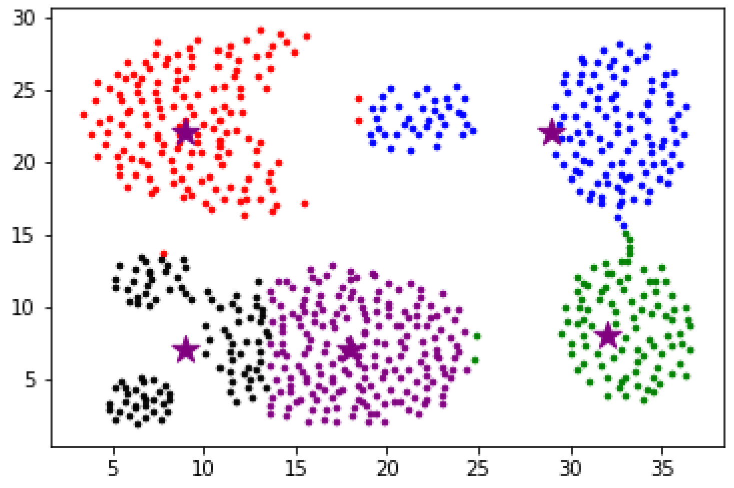
In [10]:
```
total0 = 0
total1 = 0
total2 = 0
total3 = 0
total4 = 0
for i in range (clust):
    total0 = num.sum([hitdistance(dt1[j],pxy[i],a=0)for j in range (len
(dt1)) if k[j]==i])
    total1 = num.sum([hitdistance(dt1[j],pxy[i],a=0)for j in range (len
(dt1)) if k[j]==i])
    total2 = num.sum([hitdistance(dt1[j],pxy[i],a=0)for j in range (len
(dt1)) if k[j]==i])
    total3 = num.sum([hitdistance(dt1[j],pxy[i],a=0)for j in range (len
(dt1)) if k[j]==i])
    total4 = num.sum([hitdistance(dt1[j],pxy[i],a=0)for j in range (len
(dt1)) if k[j]==i])
```

```
SSE = total0+total1+total2+total3+total4
print(SSE)
```

```
1662.364784207581
```

In [11]:
```
colors = ["purple","red","black","blue","green","yellow","grey","orang
e", "brown","cyan","magenta"]
fig, ax = plt.subplots()
for i in range(clust):
    points = num.array([dt1[j] for j in range(len(dt1)) if k[j] == i])
    ax.scatter(points[:, 0], points[:, 1], s=7, c=colors[i])
ax.scatter(pxy[:, 0], pxy[:, 1], marker='*', s=200, c='purple')
```

Out[11]:
```
<matplotlib.collections.PathCollection at 0xb7e8350>
```



Kesimpulan 1:
setelah melakukan pengamatan dari hasil percobaan tersebut didimpulkan bahwa percobaan ini menghasilkan nilai Elbow ketika berada pada nilai cluster 5 dengan SSE sebesar 2464.984116 dengan menggunakan Elbow Method(untuk hasil pengamatan terlampir pada file excel).

PENGUJIAN DATA TEST

Berdasarkan data test yang dihasilkan pada pengamatan data train yaitu pada cluster 5, maka akan diperoleh tipe cluster data dan nilai SSE pada test tersebut.

Load Data Test

In [12]:
```
dtest = pd.read_csv("TestsetTugas2.txt",delimiter="\t")
print(dtest)
```

```
       n1     n2
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
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
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 x 2 columns]
```

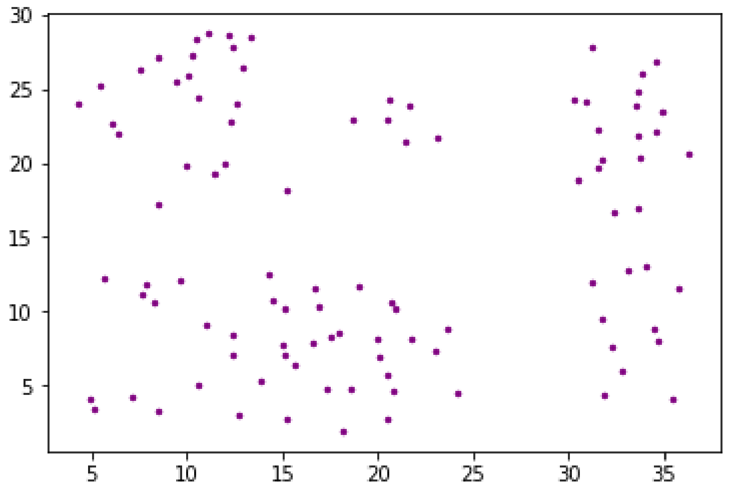
In [13]:

```
c = dtest["n1"].values
d = dtest["n2"].values
```

Visualisasi Data Test

In [14]:
```
dt2 = num.array(list(zip(c,d)))
plt.scatter(c,d, c="purple",s=6)
```

Out[14]:
```
<matplotlib.collections.PathCollection at 0xb7e8650>
```



Clustering Data Test

In [15]:
```
clust = 5

px = num.array([12,27,9,29,13])
py = num.array([4,7,22,22,10])
pxy = num.array(list(zip(px,py)))
print(pxy)
```

```
[[12  4]
 [27  7]
 [ 9 22]
 [29 22]
 [13 10]]
```

In [16]:
```
for i in range (len(dt2)):
        dnew = hitdistance(dt2[i],pxy)
        win = num.argmin(dnew)
        k[i]= win
```

In [17]:
```
for i in range (len(dt2)):
    print(i,k[i])
```

```
0 2.0
1 3.0
2 3.0
3 3.0
4 3.0
5 3.0
6 2.0
7 2.0
8 2.0
9 2.0
10 2.0
11 2.0
12 2.0
13 2.0
14 2.0
15 2.0
16 2.0
17 2.0
18 2.0
19 2.0
20 2.0
21 2.0
22 2.0
23 2.0
24 2.0
25 2.0
26 2.0
27 2.0
28 2.0
29 1.0
30 1.0
```

```
31 1.0
32 1.0
33 1.0
34 1.0
35 1.0
36 1.0
37 1.0
38 1.0
39 1.0
40 4.0
41 4.0
42 4.0
43 4.0
44 4.0
45 4.0
46 4.0
47 4.0
48 0.0
49 0.0
50 0.0
51 4.0
52 0.0
53 0.0
54 0.0
55 0.0
56 1.0
57 0.0
58 4.0
59 1.0
60 1.0
61 4.0
62 4.0
63 1.0
64 1.0
65 1.0
66 4.0
67 4.0
68 1.0
69 1.0
```

```
70 1.0
71 1.0
72 1.0
73 0.0
74 0.0
75 0.0
76 0.0
77 3.0
78 3.0
79 3.0
80 3.0
81 3.0
82 3.0
83 3.0
84 3.0
85 3.0
86 3.0
87 3.0
88 3.0
89 3.0
90 3.0
91 3.0
92 3.0
93 3.0
94 3.0
95 4.0
96 4.0
97 4.0
98 4.0
99 4.0
```

In [18]:
```
cl0 = []
cl1 = []
cl2 = []
cl3 = []
cl4 = []
for i in range (len(dt2)):
    if k[i] == 0.0:
        cl0.append(i)
    elif k[i] == 1.0:
        cl1.append(i)
    elif k[i] == 2.0:
        cl2.append(i)
    elif k[i] == 3.0:
        cl3.append(i)
    elif k[i] == 4.0:
        cl4.append(i)

print("cluster 0")
print(cl0)
print("cluster 1")
print(cl1)
print("cluster 2")
print(cl2)
print("cluster 3")
print(cl3)
print("cluster 4")
print(cl4)
```

```
cluster 0
[48, 49, 50, 52, 53, 54, 55, 57, 73, 74, 75, 76]
cluster 1
[29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 56, 59, 60, 63, 64, 65, 6
8, 69, 70, 71, 72]
cluster 2
[0, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25, 26, 27, 28]
cluster 3
[1, 2, 3, 4, 5, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90,
91, 92, 93, 94]
cluster 4
[40, 41, 42, 43, 44, 45, 46, 47, 51, 58, 61, 62, 66, 67, 95, 96, 97, 9
8, 99]
```

Menghitung Quality of Cluster dengan SSE

In [19]:
```
total0 = 0
```

```
total1 = 0
total2 = 0
total3 = 0
total4 = 0
for i in range (clust):
    total0 = num.sum([hitdistance(dt2[j],pxy[i],a=0)for j in range (len
(dt2)) if k[j]==i])
    total1 = num.sum([hitdistance(dt2[j],pxy[i],a=0)for j in range (len
(dt2)) if k[j]==i])
    total2 = num.sum([hitdistance(dt2[j],pxy[i],a=0)for j in range (len
(dt2)) if k[j]==i])
    total3 = num.sum([hitdistance(dt2[j],pxy[i],a=0)for j in range (len
(dt2)) if k[j]==i])
    total4 = num.sum([hitdistance(dt2[j],pxy[i],a=0)for j in range (len
(dt2)) if k[j]==i])

SSE = total0+total1+total2+total3+total4
print(SSE)
```

```
377.7429780020818
```

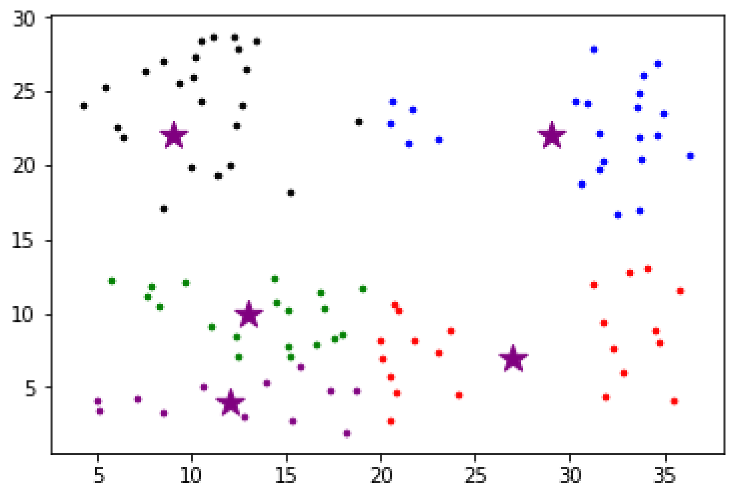
Visualisasi Clustering

In [20]:

```
colors = ["purple","red","black","blue","green","yellow","grey","orang
e", "brown"]
fig, ax = plt.subplots()
for i in range(clust):
    points = num.array([dt2[j] for j in range(len(dt2)) if k[j] == i])
    ax.scatter(points[:, 0], points[:, 1], s=7, c=colors[i])
ax.scatter(pxy[:, 0], pxy[:, 1], marker='*', s=200, c='purple')
```

Out[20]:

```
<matplotlib.collections.PathCollection at 0xb8880f0>
```



Kesimpulan 2 : Dengan melakukan pengujian dari data test pada nilai cluster 5 didapatkan nilai centroidnya 377.7429780020818

Kesimpulan

Dapat disimpulkan bahwa hasil observasi data train yang digunakan pada data test, membuat kualitas dari nilai cluster dengan centroid tersebut baik dengan nilai SSE yang terbilang kecil.