I implemented the k means algorithm step by step. I started simple by doing only one initialization and then expanded it with a loop and at last put everything into a function. My main kmeans function returns not only the best clusters and final centroids, the wcss error, but also it returns all the saved steps that makes it easy to plot all the things.

## K =3 , inits (r) =10

First, I report 10 different inits (r) ,  and wcss error for each of them + the pattern of loss changes in each iteration.

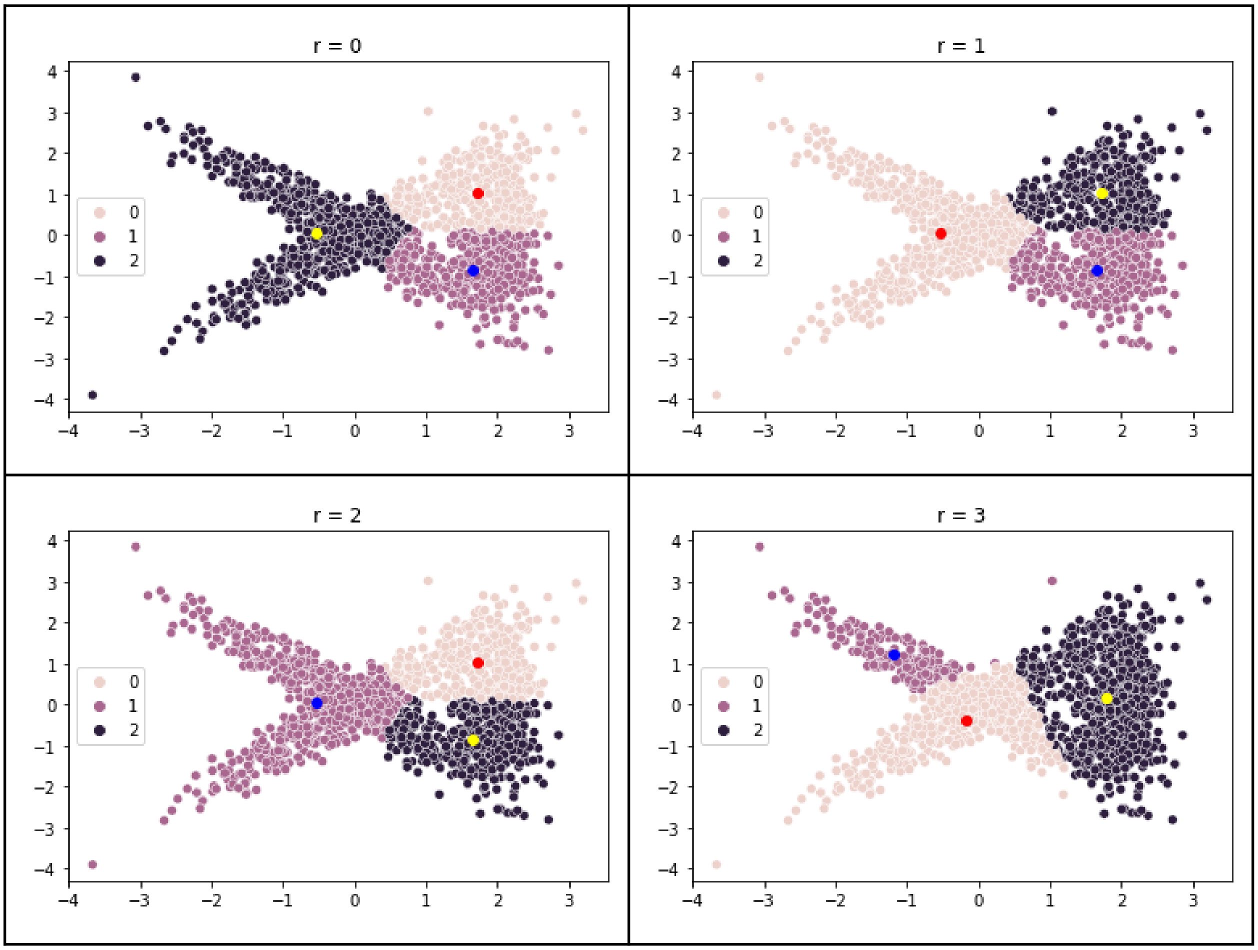
```
----------------- r = 0----------------
iteration:  1 loss:  1123
iteration:  2 loss:  69
iteration:  3 loss:  17
iteration:  4 loss:  3
iteration:  5 loss:  0
error 1539.2773846088364
----------------- r = 1----------------
iteration:  1 loss:  942
iteration:  2 loss:  225
iteration:  3 loss:  117
iteration:  4 loss:  53
iteration:  5 loss:  20
iteration:  6 loss:  9
iteration:  7 loss:  7
iteration:  8 loss:  5
iteration:  9 loss:  1
iteration:  10 loss:  0
error 1539.2773846088405
----------------- r = 2----------------
iteration:  1 loss:  1045
iteration:  2 loss:  75
iteration:  3 loss:  52
iteration:  4 loss:  72
iteration:  5 loss:  109
iteration:  6 loss:  114
iteration:  7 loss:  101
iteration:  8 loss:  57
iteration:  9 loss:  23
iteration:  10 loss:  8
iteration:  11 loss:  7
iteration:  12 loss:  1
iteration:  13 loss:  1
iteration:  14 loss:  0
error 1539.2773846088382
----------------- r = 3----------------
```

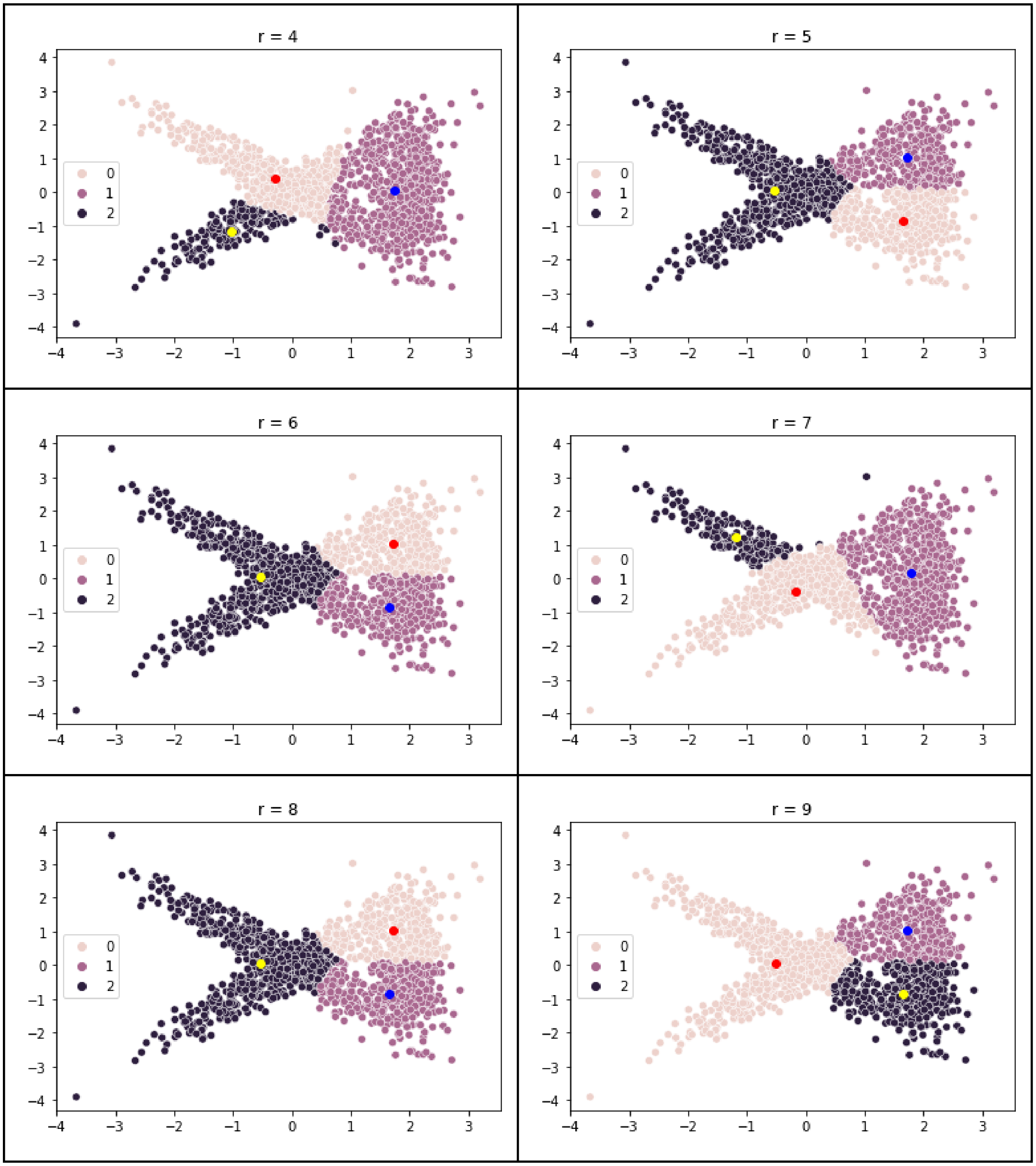
```
iteration:  1 loss:  1163
iteration:  2 loss:  291
iteration:  3 loss:  214
iteration:  4 loss:  66
iteration:  5 loss:  37
iteration:  6 loss:  39
iteration:  7 loss:  51
iteration:  8 loss:  42
iteration:  9 loss:  45
iteration:  10 loss:  33
iteration:  11 loss:  38
iteration:  12 loss:  33
iteration:  13 loss:  16
iteration:  14 loss:  12
iteration:  15 loss:  7
iteration:  16 loss:  4
iteration:  17 loss:  1
iteration:  18 loss:  0
error 1768.1733623909624
----------------- r = 4----------------
iteration:  1 loss:  637
iteration:  2 loss:  151
iteration:  3 loss:  58
iteration:  4 loss:  25
iteration:  5 loss:  12
iteration:  6 loss:  12
iteration:  7 loss:  2
iteration:  8 loss:  1
iteration:  9 loss:  1
iteration:  10 loss:  0
error 1846.6616151711696
----------------- r = 5----------------
iteration:  1 loss:  715
iteration:  2 loss:  126
iteration:  3 loss:  49
iteration:  4 loss:  38
iteration:  5 loss:  63
iteration:  6 loss:  79
iteration:  7 loss:  97
iteration:  8 loss:  114
iteration:  9 loss:  107
iteration:  10 loss:  72
iteration:  11 loss:  34
iteration:  12 loss:  11
```

```
iteration:  13 loss:  4
iteration:  14 loss:  1
iteration:  15 loss:  0
error 1539.2773846088362
----------------- r = 6----------------
iteration:  1 loss:  1187
iteration:  2 loss:  110
iteration:  3 loss:  71
iteration:  4 loss:  123
iteration:  5 loss:  143
iteration:  6 loss:  96
iteration:  7 loss:  30
iteration:  8 loss:  10
iteration:  9 loss:  2
iteration:  10 loss:  0
error 1539.2773846088364
----------------- r = 7----------------
iteration:  1 loss:  689
iteration:  2 loss:  127
iteration:  3 loss:  121
iteration:  4 loss:  57
iteration:  5 loss:  20
iteration:  6 loss:  9
iteration:  7 loss:  3
iteration:  8 loss:  5
iteration:  9 loss:  5
iteration:  10 loss:  1
iteration:  11 loss:  1
iteration:  12 loss:  0
error 1768.1695050741257
----------------- r = 8----------------
iteration:  1 loss:  586
iteration:  2 loss:  335
iteration:  3 loss:  178
iteration:  4 loss:  115
iteration:  5 loss:  59
iteration:  6 loss:  16
iteration:  7 loss:  8
iteration:  8 loss:  9
iteration:  9 loss:  4
iteration:  10 loss:  1
iteration:  11 loss:  0
error 1539.2773846088364
----------------- r = 9----------------
```

```
iteration:  1 loss:  705
iteration:  2 loss:  230
iteration:  3 loss:  174
iteration:  4 loss:  78
iteration:  5 loss:  26
iteration:  6 loss:  6
iteration:  7 loss:  2
iteration:  8 loss:  1
iteration:  9 loss:  1
iteration:  10 loss:  0
error 1539.2566749916375
--------------
best r is 9 with sum square error of 1539.2566749916375
--------------
```

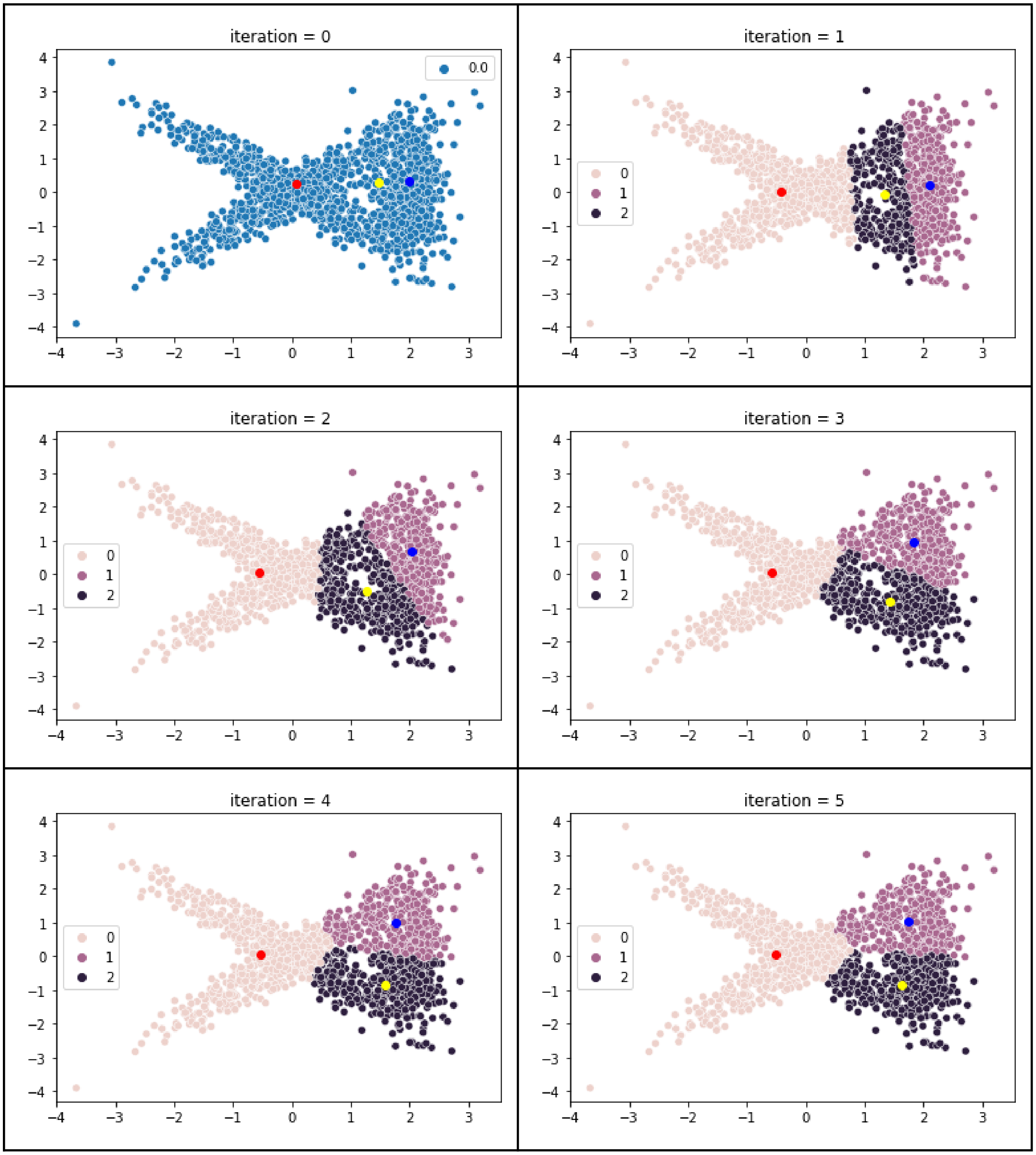
The following plots show the results for each r for k=3

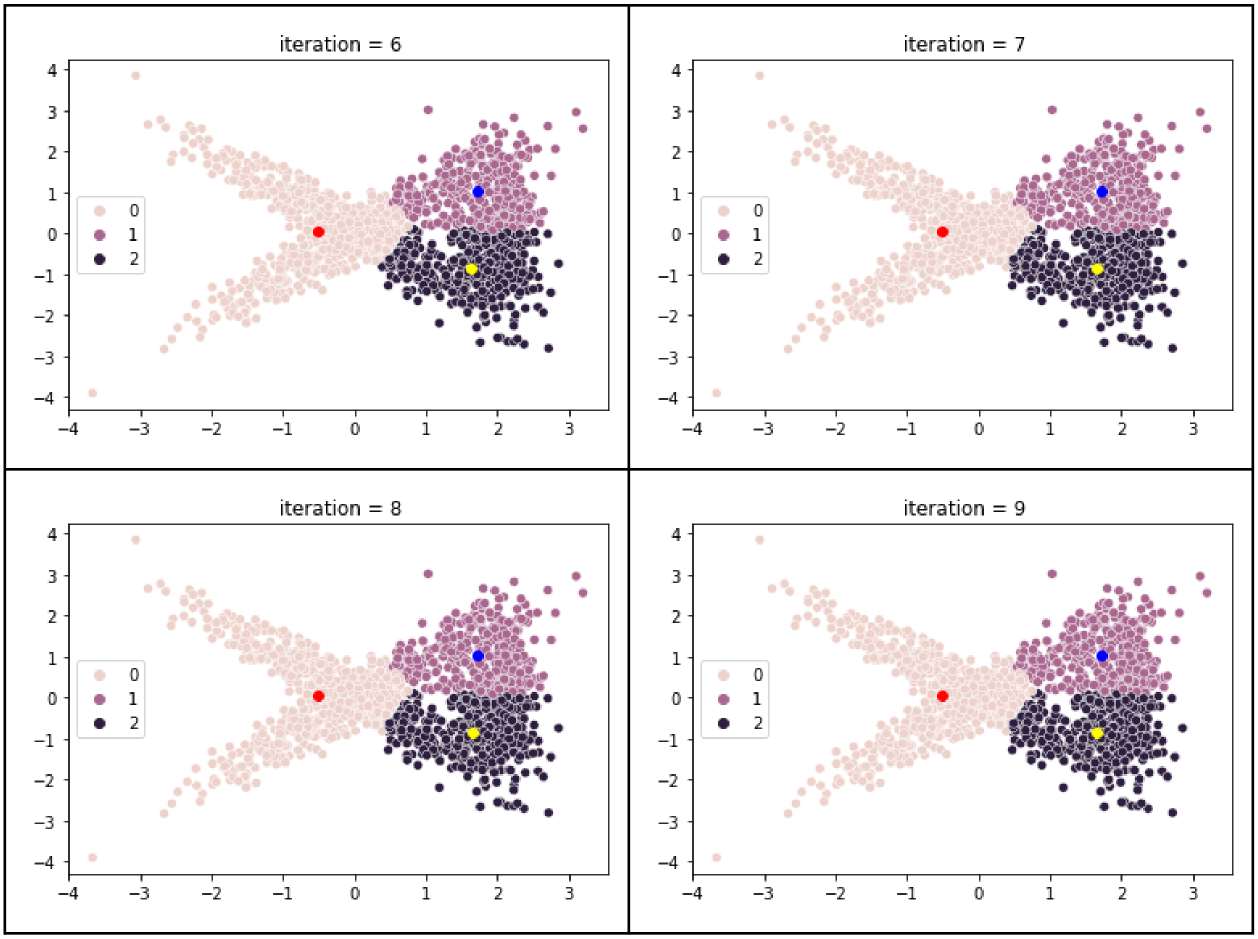




## The steps of each iteration within the best r

Second, I report the changes in cluster assignments and updates in each iteration for r=9 which was the best init for k=3. I report this just to see how my k-means converges.

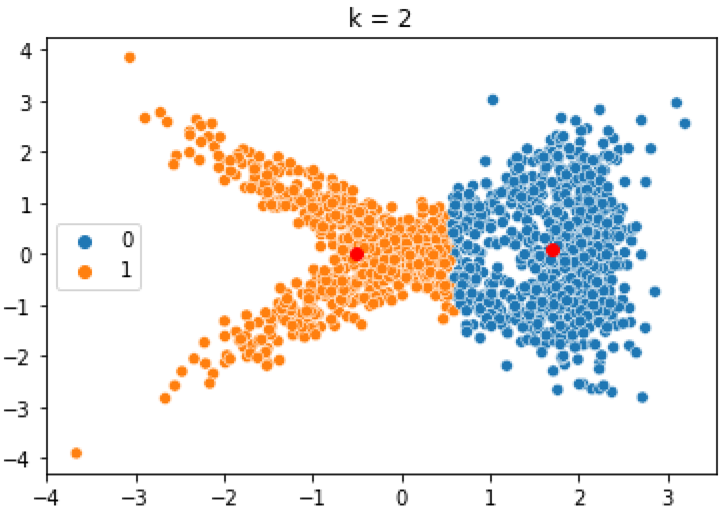

iteration = 0
0.0
iteration = 1
0
1
2
iteration = 2
0
1
2
iteration = 3
0
1
2
iteration = 4
0
1
2
iteration = 5
0
1
2
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3



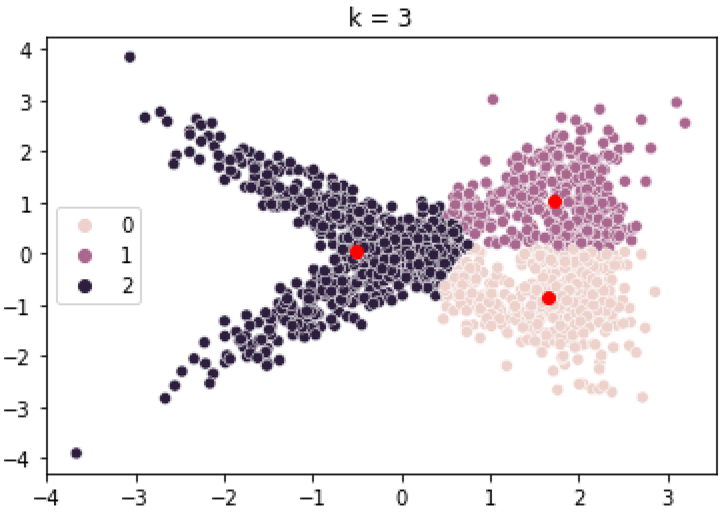
**The effect of K on clustering and wcss for each K = 2, 3, 4, and 5**

At last, here are my results for clustering with different Ks, from 2 to 5 for the best of their inits (r)
The wcss at the final step are as follows:

**K =2**
**WCSS  error of 2228.619218804034**

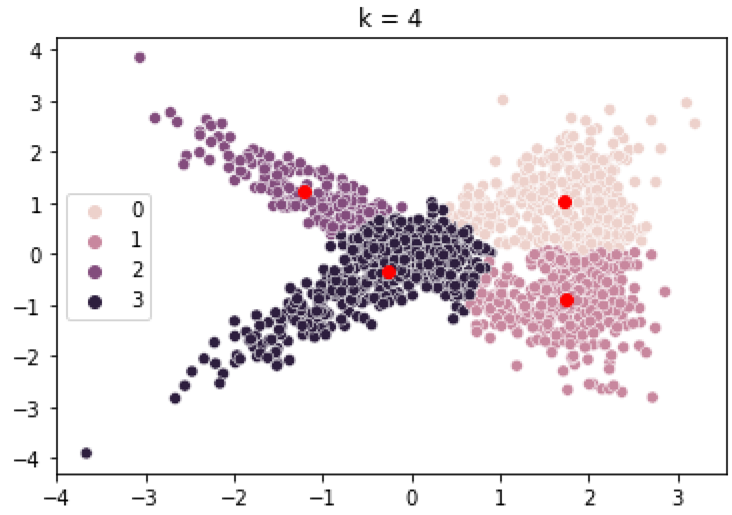


**K =3**
**WCSS error of 1539.2566749916375**



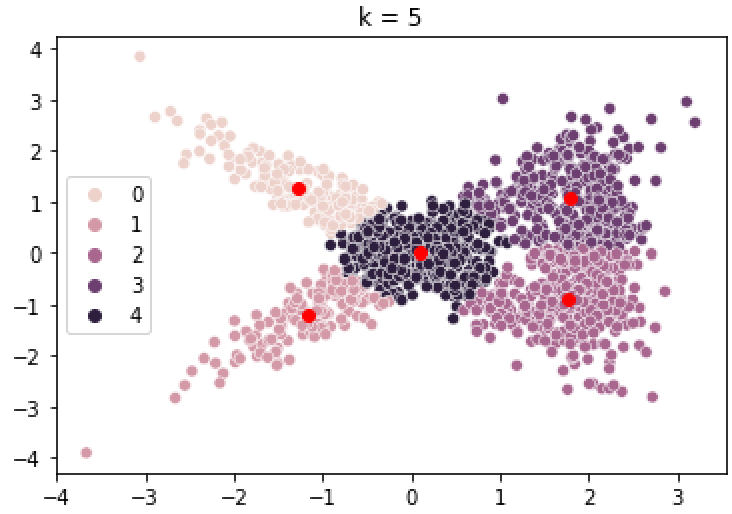
**--------------**

**K= 4**
**WCSS error of 1103.5070477213312**



--------------

**K =5**



**WCSS error of  of 773.1326716983134**

--------------

**Findings**:

1. Each random initialization may result in a different clustering and error. It is good to do it 10 times, so we can pick the best one.

2. The choice of k affects the clustering as well as the final error. The bigger the k is the lower the wcss error. Of course if we set k equal to the number of data points, there will not be a variability between data points and their clusters, so it will be zero. Bigger k does not mean better clustering.