K-means clustering is a type of unsupervised learning, which is used when you have unlabeled data (i.e., data without defined categories or groups).It allow us to find groups of similar objects, objects that are more related to each other than to objects in other groups.

Ex:

Grouping of documents, music and movies by different topics or finding customers those having similar interest based on common purchase behaviors as a basis for **recommendation engines**

One of the drawbacks of this clustering algorithm is that we have to specify the number of clusters

An inappropriate choice for clusters can result in poor clustering performance.

Algorithm

- Select k points at random as cluster centers.
- Assign objects to their closest cluster center according to the Euclidean distance function.
- Calculate the centroid or mean of all objects in each cluster.
- Repeat steps 2, 3 and 4 until the same points are assigned to each cluster in consecutive rounds.

Example-1(With 1 Input):

Suppose we want to group the visitors to a website using just their age (one-dimensional space) as follows:

15,15,16,19,19,20,20,21,22,28,35,40,41,42,43,44,60,61,65

n = 19,k=2

Sol:

Select Initial clusters with 2 any random values:

Let's

c1 = 16
c2 = 22

Iteration 1:

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	16	22	1	7	1	**(15+15+16)/3= 15.33**
15	16	22	1	7	1	
16	16	22	0	6	1	
19	16	22	9	3	2	**(19+19+20+20+21+22+28+35+40 +41+42+43+44+60+61+65)/16= 36.25**
19	16	22	9	3	2	
20	16	22	16	2	2	
20	16	22	16	2	2	
21	16	22	25	1	2	
22	16	22	36	0	2	
28	16	22	12	6	2	
35	16	22	19	13	2	
40	16	22	24	18	2	
41	16	22	25	19	2	
42	16	22	26	20	2	
43	16	22	27	21	2	
44	16	22	28	22	2	
60	16	22	44	38	2	
61	16	22	45	39	2	
65	16	22	49	43	2	

Iteration 2:

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	15.33	36.25	0.33	21.25	1	**(15+15+16+19+19+20+20+21+22)/9= 18.56**
15	15.33	36.25	0.33	21.25	1	
16	15.33	36.25	0.67	20.25	1	
19	15.33	36.25	3.67	17.25	1	
19	15.33	36.25	3.67	17.25	1	
20	15.33	36.25	4.67	16.25	1	
20	15.33	36.25	4.67	16.25	1	
21	15.33	36.25	5.67	15.25	1	
22	15.33	36.25	6.67	14.25	1	
28	15.33	36.25	12.67	8.25	2	**(28+35+40+41+42+43+44+60+61+65)/10 =45.9**
35	15.33	36.25	19.67	1.25	2	
40	15.33	36.25	24.67	3.75	2	
41	15.33	36.25	25.67	4.75	2	
42	15.33	36.25	26.67	5.75	2	
43	15.33	36.25	27.67	6.75	2	
44	15.33	36.25	28.67	7.75	2	
60	15.33	36.25	44.67	23.75	2	
61	15.33	36.25	45.67	24.75	2	
65	15.33	36.25	49.67	28.75	2	

Iteration 3:

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	18.56	45.9	3.56	30.9	1	**19.50**
15	18.56	45.9	3.56	30.9	1	
16	18.56	45.9	2.56	29.9	1	
19	18.56	45.9	0.44	26.9	1	
19	18.56	45.9	0.44	26.9	1	
20	18.56	45.9	1.44	25.9	1	
20	18.56	45.9	1.44	25.9	1	
21	18.56	45.9	2.44	24.9	1	
22	18.56	45.9	3.44	23.9	1	
28	18.56	45.9	9.44	17.9	1	
35	18.56	45.9	16.44	10.9	2	**47.89**
40	18.56	45.9	21.44	5.9	2	
41	18.56	45.9	22.44	4.9	2	
42	18.56	45.9	23.44	3.9	2	
43	18.56	45.9	24.44	2.9	2	
44	18.56	45.9	25.44	1.9	2	
60	18.56	45.9	41.44	14.1	2	
61	18.56	45.9	42.44	15.1	2	
65	18.56	45.9	46.44	19.1	2	

Iteration 4:

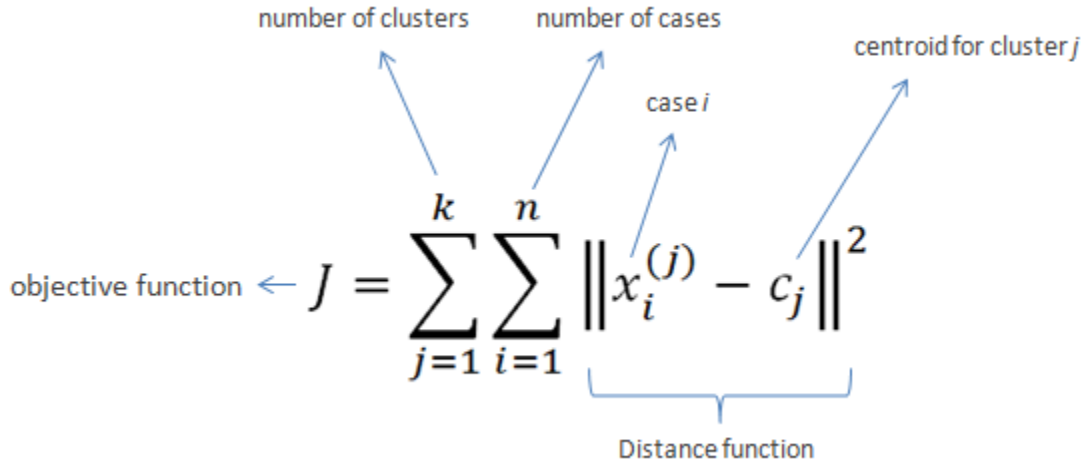
x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	19.5	47.89	4.50	32.89	1	
15	19.5	47.89	4.50	32.89	1	
16	19.5	47.89	3.50	31.89	1	
19	19.5	47.89	0.50	28.89	1	
19	19.5	47.89	0.50	28.89	1	**19.50**
20	19.5	47.89	0.50	27.89	1	
20	19.5	47.89	0.50	27.89	1	
21	19.5	47.89	1.50	26.89	1	
22	19.5	47.89	2.50	25.89	1	
28	19.5	47.89	8.50	19.89	1	
35	19.5	47.89	15.50	12.89	2	
40	19.5	47.89	20.50	7.89	2	
41	19.5	47.89	21.50	6.89	2	
42	19.5	47.89	22.50	5.89	2	
43	19.5	47.89	23.50	4.89	2	**47.89**
44	19.5	47.89	24.50	3.89	2	
60	19.5	47.89	40.50	12.11	2	
61	19.5	47.89	41.50	13.11	2	
65	19.5	47.89	45.50	17.11	2	

No change between iterations 3 and 4 has been noted. By using clustering, 2 groups have been identified 15-28 and 35-65. The initial choice of centroids can affect the output clusters, so the algorithm is often run multiple times with different starting conditions in order to get a fair view what the clusters should be.

we set n_init=5 to run the k-means clustering algorithm 5 times independently with different random initial centroids to choose the final model as the one with the lowest SSE.

km = KMeans(n_clusters=2,n_init=5, **max_iter =100**)

It means for 2 clusters it will run 5 times with different randomly selected initial centroids with 100 iteration each run and internally compute SSE and the lowest SSE centriods considered as final centroids.



A good clustering algorithm having low with-in cluster sum of square error

We may directly find centeriods having lowest SSE in Code:

Km. cluster_centers_

And

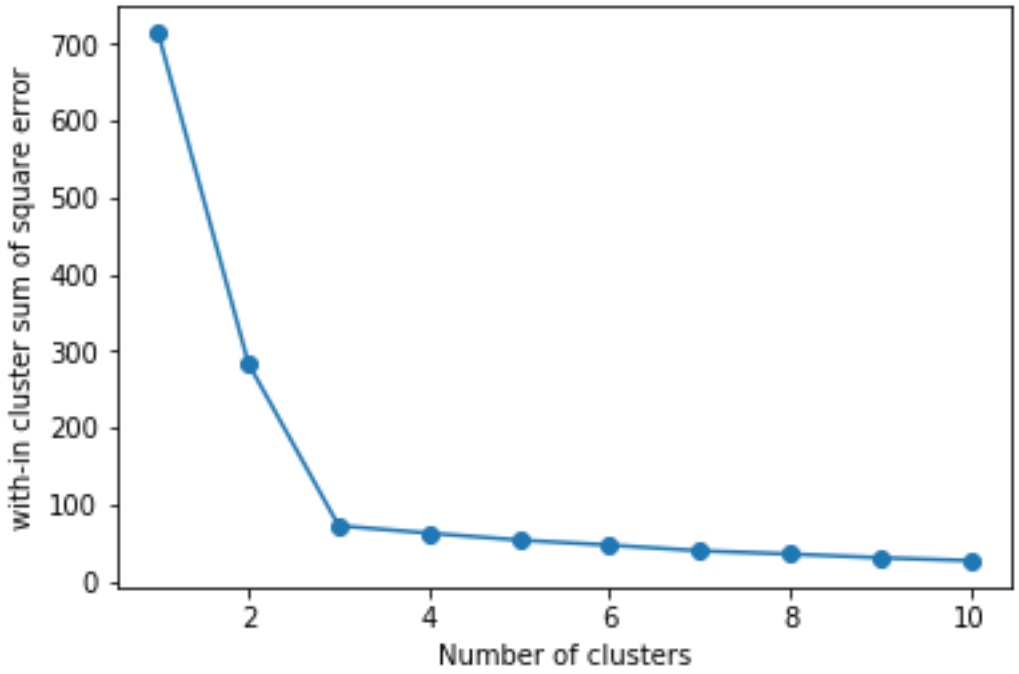
Km. inertia_

Now the question is how to find optimal number of clusters…?

Sol:

Elbow method

```
WSSE = []

for i in range(1,11):

    km = KMeans(n_clusters=i)

    km.fit(X)

    WSSE.append(km.inertia_)

plt.plot(range(1,11),WSSE,marker='o')

plt.xlabel('Number of clusters')

plt.ylabel('with-in cluster sum of square error')

plt.show()
```



the optimum clusters is where the elbow occurs(k=3).

Silhouette Analysis

Mean-shift clustering

Example 2 with 2 Inputs:

As a simple illustration of a k-means algorithm, consider the following data set consisting of the scores of two variables on each of seven individuals:

Subject	A	B
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

This data set is to be grouped into two clusters. As a first step in finding a sensible initial partition, let the A & B values of the two individuals furthest apart (using the Euclidean distance measure), define the initial cluster means, giving:

	Individual	(Centroids)
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

Calculate distance of each individual with centroids:

Individual	Centroid 1	Centroid 2
1	0	7.21
2 (1.5, 2.0)	1.12	6.10
3	3.61	3.61
4	7.21	0
5	4.72	2.5
6	5.31	2.06
7	4.30	2.92

Step 2:

- Thus, we obtain two clusters containing:

  {1,2,3} and {4,5,6,7}.
- Their new centroids are:

$$m_1 = \left(\frac{1}{3}(1.0+1.5+3.0), \frac{1}{3}(1.0+2.0+4.0)\right) = (1.83, 2.33)$$

$$m_2 = \left(\frac{1}{4}(5.0+3.5+4.5+3.5), \frac{1}{4}(7.0+5.0+5.0+4.5)\right)$$

$$= (4.12, 5.38)$$

Step 3:

- Now using these centroids we compute the Euclidean distance of each object, as shown in table.

Individual	Centroid 1	Centroid 2
1	1.57	5.38
2	0.47	4.28
3	2.04	1.78
4	5.64	1.84
5	3.15	0.73
6	3.78	0.54
7	2.74	1.08

- Therefore, the new clusters are:
  {1,2} and {3,4,5,6,7}

- Next centroids are:
  m1=(1.25,1.5) and m2 = (3.9,5.1)

Again Compute Distance and form clusters

- Step 4 :

  The clusters obtained are:

  {1,2} and {3,4,5,6,7}

- Therefore, there is no change in the cluster.
- Thus, the algorithm comes to a halt here and final result consist of 2 clusters {1,2} and {3,4,5,6,7}.