*Class.*sklearn.cluster.KMeans(*n_clusters=8,*,init='k-means++',n_init=10,max_iter=300,tol=0.0001,precompute_distances='deprecated',verbose=0,random_state=None,copy_x=True,n_jobs='deprecated',algorithm='auto'*)

n_clusters*int, default=8*
The number of clusters to form as well as the number of centroids to generate.

init*{'k-means++', 'random'}, callable or array-like of shape (n_clusters, n_features), default='k-means++'*

Method for initialization:

'k-means++' : selects initial cluster centers for k-mean clustering in a smart way to speed up convergence. See section Notes in k_init for more details.

'random': choose `n_clusters` observations (rows) at random from data for the initial centroids.

If an array is passed, it should be of shape (n_clusters, n_features) and gives the initial centers.

If a callable is passed, it should take arguments X, n_clusters and a random state and return an initialization.

n_init*int, default=10*

Number of time the k-means algorithm will be run with different centroid seeds. The final results will be the best output of n_init consecutive runs in terms of inertia.

max_iter*int, default=300*

Maximum number of iterations of the k-means algorithm for a single run.

tol*float, default=1e-4*

Relative tolerance with regards to Frobenius norm of the difference in the cluster centers of two consecutive iterations to declare convergence.

precompute_distances*{'auto', True, False}, default='auto'*

Precompute distances (faster but takes more memory).

'auto' : do not precompute distances if n_samples * n_clusters > 12 million. This corresponds to about 100MB overhead per job using double precision.

True : always precompute distances.

False : never precompute distances.

Deprecated since version 0.23: 'precompute_distances' was deprecated in version 0.22 and will be removed in 1.0 (renaming of 0.25). It has no effect.

verbose*int, default=0*

Verbosity mode.

random_state*int, RandomState instance or None, default=None*

Determines random number generation for centroid initialization. Use an int to make the randomness deterministic. See Glossary.

copy_x*bool, default=True*

When pre-computing distances it is more numerically accurate to center the data first. If copy_x is True (default), then the original data is not modified. If False, the original data is modified, and put back before the function returns, but small numerical differences may be introduced by subtracting and then adding the data mean. Note that if the original data is not C-contiguous, a copy will be made even if copy_x is False. If the original data is sparse, but not in CSR format, a copy will be made even if copy_x is False.

n_jobs*int, default=None*

The number of OpenMP threads to use for the computation. Parallelism is sample-wise on the main cython loop which assigns each sample to its closest center.

`None` or `-1` means using all processors.

Deprecated since version 0.23: `n_jobs` was deprecated in version 0.23 and will be removed in 1.0 (renaming of 0.25).

algorithm*{“auto”, “full”, “elkan”}, default=”auto”*

K-means algorithm to use. The classical EM-style algorithm is “full”. The “elkan” variation is more efficient on data with well-defined clusters, by using the triangle inequality. However it’s more memory intensive due to the allocation of an extra array of shape (n_samples, n_clusters).

For now “auto” (kept for backward compatibiliy) chooses “elkan” but it might change in the future for a better heuristic.