*class* sklearn.cluster.**KMeans**(*n\_clusters=8*, *init=’k-means++’*, *n\_init=10*, *max\_iter=300*, *tol=0.0001*, *precompute\_distances=’auto’*, *verbose=0*, *random\_state=None*, *copy\_x=True*, *n\_jobs=1*, *algorithm=’auto’*)[[source]](https://github.com/scikit-learn/scikit-learn/blob/ef5cb84a/sklearn/cluster/k_means_.py#L707)[¶](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans)

K-Means clustering

Read more in the [User Guide](http://scikit-learn.org/stable/modules/clustering.html#k-means).

|  |  |
| --- | --- |
| **Parameters:** | **n\_clusters** : int, optional, default: 8  The number of clusters to form as well as the number of centroids to generate.  **init** : {‘k-means++’, ‘random’ or an ndarray}  Method for initialization, defaults to ‘k-means++’:  ‘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 k observations (rows) at random from data for the initial centroids.  If an ndarray is passed, it should be of shape (n\_clusters, n\_features) and gives the initial centers.  **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 inertia to declare convergence  **precompute\_distances** : {‘auto’, True, False}  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  **verbose** : int, default 0  Verbosity mode.  **random\_state** : int, RandomState instance or None, optional, default: None  If int, random\_state is the seed used by the random number generator; If RandomState instance, random\_state is the random number generator; If None, the random number generator is the RandomState instance used by *np.random*.  **copy\_x** : boolean, default True  When pre-computing distances it is more numerically accurate to center the data first. If copy\_x is True, 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.  **n\_jobs** : int  The number of jobs to use for the computation. This works by computing each of the n\_init runs in parallel.  If -1 all CPUs are used. If 1 is given, no parallel computing code is used at all, which is useful for debugging. For n\_jobs below -1, (n\_cpus + 1 + n\_jobs) are used. Thus for n\_jobs = -2, all CPUs but one are used.  **algorithm** : “auto”, “full” or “elkan”, default=”auto”  K-means algorithm to use. The classical EM-style algorithm is “full”. The “elkan” variation is more efficient by using the triangle inequality, but currently doesn’t support sparse data. “auto” chooses “elkan” for dense data and “full” for sparse data. |
| **Attributes:** | **cluster\_centers\_** : array, [n\_clusters, n\_features]  Coordinates of cluster centers  **labels\_ :** :  Labels of each point  **inertia\_** : float  Sum of distances of samples to their closest cluster center. |

See also

[**MiniBatchKMeans**](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html#sklearn.cluster.MiniBatchKMeans)

Alternative online implementation that does incremental updates of the centers positions using mini-batches. For large scale learning (say n\_samples > 10k) MiniBatchKMeans is probably much faster than the default batch implementation.

**Notes**

The k-means problem is solved using Lloyd’s algorithm.

The average complexity is given by O(k n T), were n is the number of samples and T is the number of iteration.

The worst case complexity is given by O(n^(k+2/p)) with n = n\_samples, p = n\_features. (D. Arthur and S. Vassilvitskii, ‘How slow is the k-means method?’ SoCG2006)

In practice, the k-means algorithm is very fast (one of the fastest clustering algorithms available), but it falls in local minima. That’s why it can be useful to restart it several times.

**Examples**

>>>

**>>> from** **sklearn.cluster** **import** KMeans

**>>> import** **numpy** **as** **np**

**>>>** X = np.array([[1, 2], [1, 4], [1, 0],

**...**  [4, 2], [4, 4], [4, 0]])

**>>>** kmeans = KMeans(n\_clusters=2, random\_state=0).fit(X)

**>>>** kmeans.labels\_

array([0, 0, 0, 1, 1, 1], dtype=int32)

**>>>** kmeans.predict([[0, 0], [4, 4]])

array([0, 1], dtype=int32)

**>>>** kmeans.cluster\_centers\_

array([[ 1., 2.],

[ 4., 2.]])

**Methods**

|  |  |
| --- | --- |
| [**fit**](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.fit)(X[, y]) | Compute k-means clustering. |
| [**fit\_predict**](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.fit_predict)(X[, y]) | Compute cluster centers and predict cluster index for each sample. |
| [**fit\_transform**](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.fit_transform)(X[, y]) | Compute clustering and transform X to cluster-distance space. |
| [**get\_params**](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.get_params)([deep]) | Get parameters for this estimator. |
| [**predict**](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.predict)(X) | Predict the closest cluster each sample in X belongs to. |
| [**score**](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.score)(X[, y]) | Opposite of the value of X on the K-means objective. |
| [**set\_params**](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.set_params)(\*\*params) | Set the parameters of this estimator. |
| [**transform**](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.transform)(X) | Transform X to a cluster-distance space. |

**\_\_init\_\_**(*n\_clusters=8*, *init=’k-means++’*, *n\_init=10*, *max\_iter=300*, *tol=0.0001*, *precompute\_distances=’auto’*, *verbose=0*, *random\_state=None*, *copy\_x=True*, *n\_jobs=1*, *algorithm=’auto’*)[[source]](https://github.com/scikit-learn/scikit-learn/blob/ef5cb84a/sklearn/cluster/k_means_.py#L839)[¶](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.__init__)

**fit**(*X*, *y=None*)[[source]](https://github.com/scikit-learn/scikit-learn/blob/ef5cb84a/sklearn/cluster/k_means_.py#L875)[¶](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.fit)

Compute k-means clustering.

|  |  |
| --- | --- |
| **Parameters:** | **X** : array-like or sparse matrix, shape=(n\_samples, n\_features)  Training instances to cluster. |

**fit\_predict**(*X*, *y=None*)[[source]](https://github.com/scikit-learn/scikit-learn/blob/ef5cb84a/sklearn/cluster/k_means_.py#L896)[¶](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.fit_predict)

Compute cluster centers and predict cluster index for each sample.

Convenience method; equivalent to calling fit(X) followed by predict(X).

|  |  |
| --- | --- |
| **Parameters:** | **X** : {array-like, sparse matrix}, shape = [n\_samples, n\_features]  New data to transform. |
| **Returns:** | **labels** : array, shape [n\_samples,]  Index of the cluster each sample belongs to. |

**fit\_transform**(*X*, *y=None*)[[source]](https://github.com/scikit-learn/scikit-learn/blob/ef5cb84a/sklearn/cluster/k_means_.py#L914)[¶](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.fit_transform)

Compute clustering and transform X to cluster-distance space.

Equivalent to fit(X).transform(X), but more efficiently implemented.

|  |  |
| --- | --- |
| **Parameters:** | **X** : {array-like, sparse matrix}, shape = [n\_samples, n\_features]  New data to transform. |
| **Returns:** | **X\_new** : array, shape [n\_samples, k]  X transformed in the new space. |

**get\_params**(*deep=True*)[[source]](https://github.com/scikit-learn/scikit-learn/blob/ef5cb84a/sklearn/base.py#L212)[¶](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.get_params)

Get parameters for this estimator.

|  |  |
| --- | --- |
| **Parameters:** | **deep** : boolean, optional  If True, will return the parameters for this estimator and contained subobjects that are estimators. |
| **Returns:** | **params** : mapping of string to any  Parameter names mapped to their values. |

**predict**(*X*)[[source]](https://github.com/scikit-learn/scikit-learn/blob/ef5cb84a/sklearn/cluster/k_means_.py#L962)[¶](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.predict)

Predict the closest cluster each sample in X belongs to.

In the vector quantization literature, *cluster\_centers\_* is called the code book and each value returned by *predict* is the index of the closest code in the code book.

|  |  |
| --- | --- |
| **Parameters:** | **X** : {array-like, sparse matrix}, shape = [n\_samples, n\_features]  New data to predict. |
| **Returns:** | **labels** : array, shape [n\_samples,]  Index of the cluster each sample belongs to. |

**score**(*X*, *y=None*)[[source]](https://github.com/scikit-learn/scikit-learn/blob/ef5cb84a/sklearn/cluster/k_means_.py#L985)[¶](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.score)

Opposite of the value of X on the K-means objective.

|  |  |
| --- | --- |
| **Parameters:** | **X** : {array-like, sparse matrix}, shape = [n\_samples, n\_features]  New data. |
| **Returns:** | **score** : float  Opposite of the value of X on the K-means objective. |

**set\_params**(*\*\*params*)[[source]](https://github.com/scikit-learn/scikit-learn/blob/ef5cb84a/sklearn/base.py#L249)[¶](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.set_params)

Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as pipelines). The latter have parameters of the form <component>\_\_<parameter> so that it’s possible to update each component of a nested object.

|  |  |
| --- | --- |
| **Returns:** | **self** : |

**transform**(*X*)[[source]](https://github.com/scikit-learn/scikit-learn/blob/ef5cb84a/sklearn/cluster/k_means_.py#L936)[¶](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.transform)

Transform X to a cluster-distance space.

In the new space, each dimension is the distance to the cluster centers. Note that even if X is sparse, the array returned by *transform* will typically be dense.

|  |  |
| --- | --- |
| **Parameters:** | **X** : {array-like, sparse matrix}, shape = [n\_samples, n\_features]  New data to transform. |
| **Returns:** | **X\_new** : array, shape [n\_samples, k]  X transformed in the new space. |