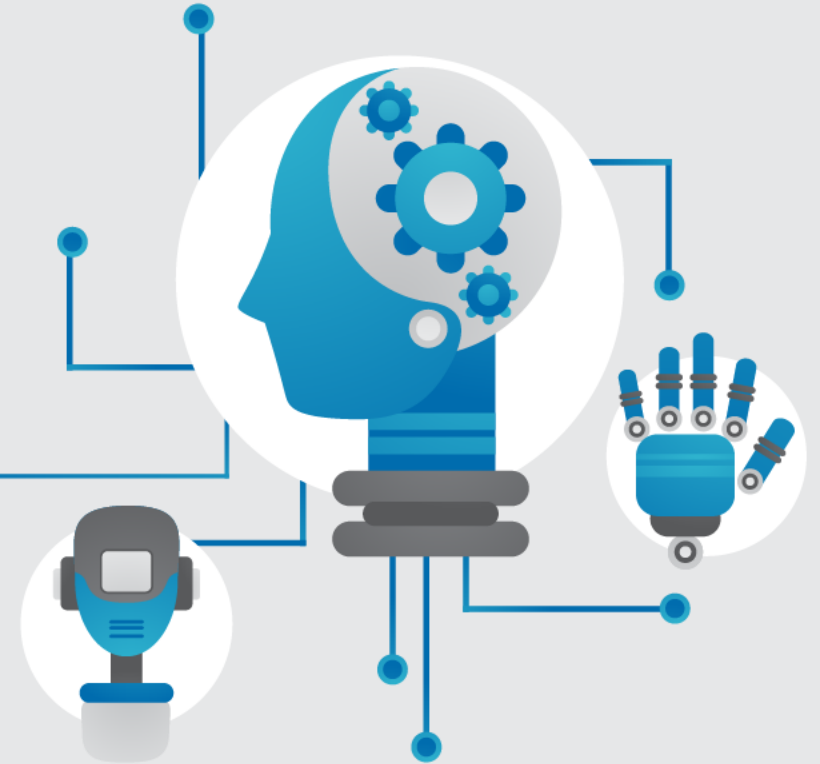


# K-means 實作






# K-means 實作

機器學習實務



› sklearn.cluster.Kmeans是K-means演算法的實作

 [Install](#) [User Guide](#) [API](#) [Examples](#) [More](#)  [Go](#)

[Prev](#) [Up](#) [Next](#)

**scikit-learn 0.22.2**  
[Other versions](#)

Please [cite us](#) if you use the software.

[sklearn.cluster.KMeans](#)  
Examples using  
[sklearn.cluster.KMeans](#)

## sklearn.cluster.KMeans

```
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=None, algorithm='auto')
```

[\[source\]](#)

K-Means clustering.

Read more in the [User Guide](#).

Parameters:	
<b>n_clusters : int, default=8</b>	The number of clusters to form as well as the number of centroids to generate.
<b>init : {'k-means++', 'random'} or ndarray of shape (n_clusters, n_features), default='k-means++'</b>	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.
<b>n_init : int, default=10</b>	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.



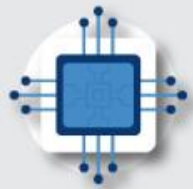
# K-means 參數說明



› `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=None,  
algorithm='auto')`

› 常用參數

- `n_clusters`
- `init`
- `n_init`
- `max_iter`
- `tol`
- `precompute_distances`
- `copy_x`
- `algorithm`



# K-means 參數說明



› **n\_clusters : int, default=8**

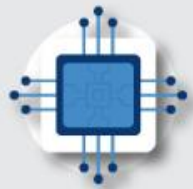
- 群組中心數量

› **Init : {'k-means++', 'random'} or ndarray of shape (n\_clusters, n\_features), default='k-means++'**

- 群組中心數量群組中心的初始化方法，有三個選擇 { 'k-means++' , 'random' or an ndarray } , 如果傳入為矩陣 ( ndarray ) , 則將該矩陣中的每一行作為群組中心

› **n\_init : int, default=10**

- k-means演算法會隨機運行n\_init次 ( 挑選不同起始中心 ) , 最終回傳最好的一個分群結果



# K-means 參數說明



## › **max\_iter : int, default=300**

- 演算法運行的最大迭代次數

## › **tol : float, default=1e-4**

- 容忍的最小誤差，當誤差小於tol就會退出迭代

## › **precompute\_distances : 'auto' or bool, default='auto'**

- 是否將數據全部事先計算查表，可選{ 'auto' , True, False}，開啟時速度更快但是更耗記憶體



# K-means 參數說明



## › **copy\_x** : bool, default=True

- 是否直接在原矩陣上進行計算。設定True，會copy一份進行計算

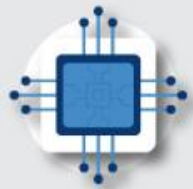
## › **algorithm** : {"auto", "full", "elkan"}, default="auto"

- K-means距離計算算法

"full" : 傳統的距離計算方式

"elkan" : 使用三角不等式，效率更高，但是目前不支持稀疏數據

"auto" : 當為稀疏矩陣時，採用full，否則elkan



# K-means 範例



```
from sklearn.cluster import KMeans
import numpy as np
X = np.array([[1, 2], [1, 4], [1, 0], [10, 2], [10, 4], [10, 0]])
kmeans = KMeans(n_clusters=2).fit(X)
print(kmeans.labels_)
```

```
In [8]: print(kmeans.labels_)
[0 0 0 1 1 1]
```

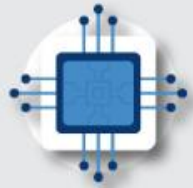
```
print(kmeans.predict([[0, 0], [12, 3]]))
```

```
In [9]: print(kmeans.predict([[0, 0], [12, 3]]))
[0 1]
```

```
print(kmeans.cluster_centers_)
```

```
In [10]: print(kmeans.cluster_centers_)
[[ 1.  2.]
 [10.  2.]
```





# 資料分群應用：Iris



```
from sklearn.datasets import load_iris
```

```
data = load_iris()
```

```
x=data['data']
```

```
y=data['target']
```

```
from sklearn.cluster import KMeans
```

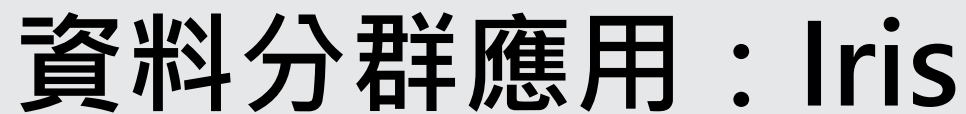
```
kmeans = KMeans(n_clusters=3).fit(x)
```

```
# 對比原本分群與kmeans結果
```

```
for i in range(3):
```

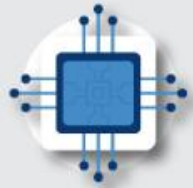
```
    print('cluster'+str(i)+': ', kmeans.labels_[y==i], end='\n\n')
```





## ➤ 分群結果

```
cluster0: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
           1 1 1 1 1 1 1 1 1 1 1 1 1]
```

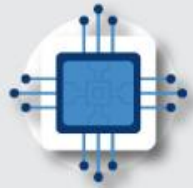


# 影像分群應用：色彩、亮度特徵分群

機器學習實務



```
import cv2, os
files=os.listdir('image')
x=[]
for file in files:
    img = cv2.imread('image/'+file)
    lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
    hist = cv2.calcHist(
        [lab],
        [0, 1, 2],
        None,
        [8,8,8], # 將3通道值的頻率統計成直方圖，每通道8個直方圖
        [0, 256, 0, 256, 0, 256])
    hist = cv2.normalize(hist, None).ravel()
    x.append(hist)
```



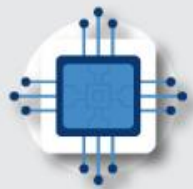
# 影像分群應用：色彩、亮度特徵分群

機器學習實務



```
from sklearn.cluster import KMeans
import numpy as np
import shutil

kmeans = KMeans(n_clusters=10).fit(x)
for i in range(10):
    os.mkdir('output/'+str(i))
    for i in range(10):
        for j in np.where(kmeans.labels_==i)[0]:
            shutil.copyfile('image/'+files[j], 'output/'+str(i)+'/'+files[j])
```



# 影像分群應用：色彩、亮度特徵分群

機器學習實務

