from sklearn.datasets import load\_iris  
from sklearn.model\_selection import cross\_val\_score  
import matplotlib.pyplot as plt  
from sklearn.neighbors import KNeighborsClassifier  
  
#读取鸢尾花数据集  
iris = load\_iris()  
x = iris.data  
y = iris.target  
k\_range = range(1, 31)  
k\_error = []  
#循环，取k=1到k=31，查看误差效果  
for k in k\_range:  
 knn = KNeighborsClassifier(n\_neighbors=k)  
 #cv参数决定数据集划分比例，这里是按照5:1划分训练集和测试集  
 scores = cross\_val\_score(knn, x, y, cv=6, scoring='accuracy')  
 k\_error.append(1 - scores.mean())  
  
#画图，x轴为k值，y值为误差值  
plt.plot(k\_range, k\_error)  
plt.xlabel('Value of K for KNN')  
plt.ylabel('Error')  
plt.show()

'''

参数解释：  
- n\_neighbors：这个值就是指 KNN 中的 “K”了。前面说到过，通过调整 K 值，算法会有不同的效果。  
- weights（权重）：最普遍的 KNN 算法无论距离如何，权重都一样，但有时候我们想搞点特殊化，比如距离更近的点让它更加重要。这时候就需要 weight 这个参数了，这个参数有三个可选参数的值，决定了如何分配权重。参数选项如下：  
 • 'uniform'：不管远近权重都一样，就是最普通的 KNN 算法的形式。  
 • 'distance'：权重和距离成反比，距离预测目标越近具有越高的权重。  
 • 自定义函数：自定义一个函数，根据输入的坐标值返回对应的权重，达到自定义权重的目的。  
- algorithm：在 sklearn 中，要构建 KNN 模型有三种构建方式，1. 暴力法，就是直接计算距离存储比较的那种放松。2. 使用 kd 树构建 KNN 模型 3. 使用球树构建。 其中暴力法适合数据较小的方式，否则效率会比较低。如果数据量比较大一般会选择用 KD 树构建 KNN 模型，而当 KD 树也比较慢的时候，则可以试试球树来构建 KNN。参数选项如下：  
 • 'brute' ：蛮力实现  
 • 'kd\_tree'：KD 树实现 KNN  
 • 'ball\_tree'：球树实现 KNN   
 • 'auto'： 默认参数，自动选择合适的方法构建模型  
不过当数据较小或比较稀疏时，无论选择哪个最后都会使用 'brute'  
   
- leaf\_size：如果是选择蛮力实现，那么这个值是可以忽略的，当使用KD树或球树，它就是是停止建子树的叶子节点数量的阈值。默认30，但如果数据量增多这个参数需要增大，否则速度过慢不说，还容易过拟合。  
- p：和metric结合使用的，当metric参数是"minkowski"的时候，p=1为曼哈顿距离， p=2为欧式距离。默认为p=2。  
- metric：指定距离度量方法，一般都是使用欧式距离。  
 • 'euclidean' ：欧式距离  
 • 'manhattan'：曼哈顿距离  
 • 'chebyshev'：切比雪夫距离  
 • 'minkowski'： 闵可夫斯基距离，默认参数  
- n\_jobs：指定多少个CPU进行运算，默认是-1，也就是全部都算。  
  
'''