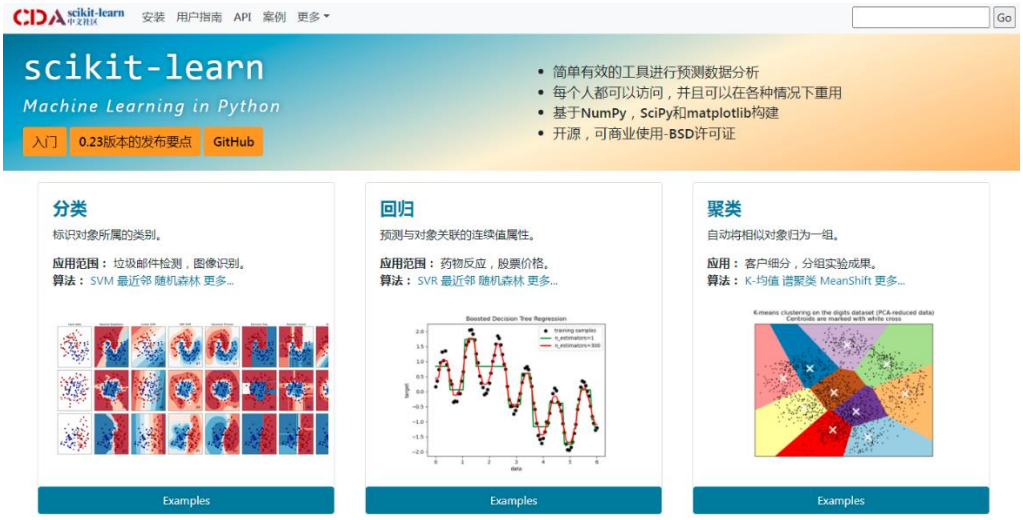
实验二 利用 scikit-learn 实现最近邻分类

准备工作（具体步骤参见实验一）：

1. 在计算机上安装好 python（推荐 3.x 版本）
2. 安装编译器和编译环境
3. 安装 scikit-learn 工具包

利用 scikit-learn 实现最近邻分类：

1. 了解 scikit-learn 工具包
   英文主页：https://scikit-learn.org/stable/
   中文社区：https://scikit-learn.org.cn/



2. 了解 scikit-learn 中的最近邻分类方法
   https://scikit-learn.org.cn/view/85.html

1.6 最近邻

`sklearn.neighbors` 提供了基于近邻的无监督和有监督的学习方法的功能。无监督最近邻是许多其他学习方法的基础，特别是流形学习(manifold learning)和光谱聚类(spectral clustering)。有监督的 neighbors-based (基于邻居的) 学习有两种方式：离散标签数据的分类和连续标签数据的回归。

最近邻方法背后的原理是找到在距离上离新样本最近的一些样本，并且从这些样本中预测标签。最近邻的样本数可以是用户定义的常数(k-最近邻)，也可以根据不同的点的局部密度(基于半径的近邻学习)确定。一般来说，距离可以用任意来度量：标准的欧氏距离是最常见的选择。基于邻居的方法被称为非泛化机器学习方法，因为它们只是"记住"它的所有训练数据(可能转换成一个快速的索引结构，比如Ball树或KD树)。

尽管它很简单，但最近邻已经成功地解决了大量的分类和回归问题，包括手写数字和卫星图像等场景。作为一种非参数方法，在决策边界非常不规则的情况下通常是成功的。

`sklearn.neighbors` 中的类输入不管是numpy中的array数组， 还是scipy.sparse矩阵都可以处理。对于密集矩阵，可以支持大量的距离度量。于稀疏矩阵，支持任意的Minkowski度量进行搜索。

有许多学习规则都是依赖于它们的核心近邻。一个例子是核密度估计(kernel density estimation,)，在密度估计(density estimation)部分讨论。

1.6.2 最近邻分类

基于近邻的分类是一种基于实例的学习或非泛化学习：它不是试图构造一个通用的内部模型，而是简单地存储训练数据的实例。分类是由每个点的最近邻的简单多数投票中计算得到的：一个查询点被标记的数据标签是由它最近邻点中最具代表性的数据标签来决定的。

scikit-learn实现了两个不同的最近邻分类器： `KNeighborsClassifier` 分类器根据每个查询点的k个最近邻实现学习，其中k是用户指定的整数值。 `RadiusNeighborsClassifier` 分类器根据每个训练点的固定半径r内的邻居数实现学习，其中 r 是用户指定的浮点值。

`KNeighborsClassifier` 中的K近邻分类是最常用的分类方法。k值的最佳选择是高度依赖于数据的：一般来说，较大的 k 会抑制噪声的影响，但使分类边界不那么清晰。

在数据不均匀采样的情况下， `RadiusNeighborsClassifier` 中基于半径的近邻分类是更好的选择。用户指定一个固定的半径r，以便在稀疏的邻居中使用更少的最近邻居来进行分类。对于高维参数空间，这种方法由于所谓的 “维度灾难” 而变得不那么有效。

基本的最近邻分类使用一样的权重：也就是说，分配给查询点的值是根据最近邻居的简单多数投票计算的。在某些情况下，最好是对邻居进行加权，这样更靠近的邻居对拟合来说贡献更大。这可以通过 `weights` 关键字来实现。默认值是 `weights="uniform"` ，为每个邻居分配同样的权重。 `weights = 'distance'` 分配的权重与到查询点的距离成反比。或者，可以提供用户定义的距离函数来计算权重.

3. 学习其中的 sklearn.neighbors.KNeighborsClassifier 方法（输入输出参数的含义、如何使用）
   https://scikit-learn.org.cn/view/695.html

sklearn.neighbors.KNeighborsClassifier

```
class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski
```

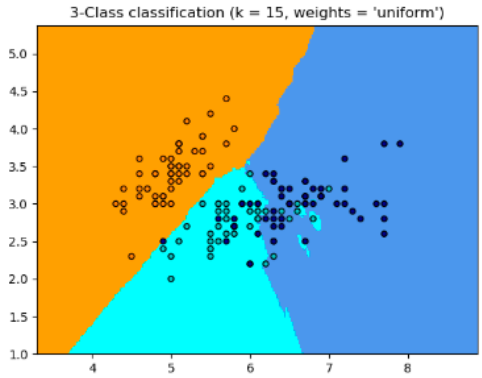
源码

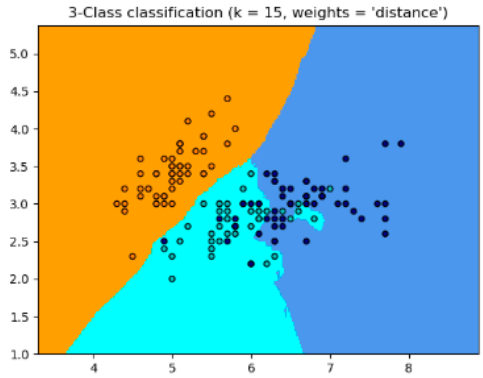
分类器执行k最近邻投票。

在用户指南中阅读更多内容。

参数	说明
n_neighbors	**int, default=5** 默认情况下用于kneighbors查询的近邻数
weights	**{'uniform', 'distance'} or callable, default='uniform'** 预测中使用的权重函数。 可能的值： "uniform"：统一权重。 每个邻域中的所有点均被加权。 "distance"：权重点与其距离的倒数。 在这种情况下，查询点的近邻比远处的近邻具有更大的影响力。 [callable]：用户定义的函数，该函数接受距离数组，并返回包含权重的相同形状的数组。
algorithm	**{'auto', 'ball_tree', 'kd_tree', 'brute'}, default='auto'** 用于计算最近临近点的算法： " ball_tree"将使用BallTree kd_tree"将使用KDTree "brute"将使用暴力搜索。 "auto"将尝试根据传递给fit方法的值来决定最合适的算法。 注意：在稀疏输入上进行拟合将使用蛮力覆盖此参数的设置。

4. 利用 KNeighborsClassifier 方法实现一个最近邻分类例子
   https://scikit-learn.org.cn/view/301.html





5. 实现第三章作业中的最近邻分类问题（可视化展示结果）

已知7个样本：X_1=(1,0), X_2=(0,1), X_3=(0,-1), X_4=(0,0), X_5=(0,2), X_6=(0,-2), X_7=(-2,0), 其中前三个为ω_1类，后四个为ω_2类。现有待分类样本X=(0.2,0.3)。

1. 编程实现k-NN分类算法，并分别计算当k=1,2,…,7时上述问题中样本X的类别。

2. 在上述问题中，当k取偶数时，可能出现两个类别中最近邻样本数相同的问题，请思考并给出此种情形下的解决方法。

6. 已知如下电影的镜头信息与其对应的电影类型：

电影名称	搞笑镜头	拥抱镜头	打斗镜头	电影类型
宝贝当家	45	2	9	喜剧片
美人鱼	21	17	5	喜剧片
澳门风云 3	54	9	11	喜剧片
功夫熊猫 3	39	0	31	喜剧片
谍影重重	5	2	57	动作片
叶问 3	3	2	65	动作片
伦敦陷落	2	3	55	动作片
我的特工爷爷	6	4	21	动作片
奔爱	7	46	4	爱情片
夜孔雀	9	39	8	爱情片
代理情人	9	38	2	爱情片
新步步惊心	8	34	17	爱情片

问题：对于电影《唐人街探案》，其包含搞笑镜头 23 个、拥抱镜头 6 个，打斗镜头 21 个，用近邻法判断其所属的电影类型（可视化结果，并讨论 K 值对结果的影响）。