第5章 分 类

对于分类问题，其实谁都不会陌生，日常生活中我们每天时刻都在进行着分类。例如，当你看到一个人，你的脑子下意识会判断他是学生，还是社会上的人；你走在路上可能会对身旁的朋友说“这个人一看就很有修养”之类的话，其实这就是一种分类操作。

分类方法是一种对离散型随机变量建模或预测的有监督学习方法。其中分类学习的目的是从给定的人工标注的分类训练样本数据集中学习出一个分类函数或者分类模型，常称为分类器(classifier)。当新的未知类别数据到来时，可以根据这个分类模型进行预测，将新数据项映射到给定类别中的某一个类中。比如说一个手写数字集保存了很多人手写的数字，通过该手写数字集训练一个分类模型，当输入一个新的未知的手写数字时，分类模型能识别出该手写数字是几。

5.1 分类的原理和步骤概述

对于分类，输入的训练数据包含两部分信息，即特征属性集(features)和类别属性(class)，类别又称为标签(label)，具体可表示为$(X_1, X_2, \cdots, X_n; y)$。其中特征属性集是指机器学习模型处理的对象或事件中收集到的已经量化的特征的集合，通常将它表示成一个向量$\boldsymbol{X} \in \mathbf{R}^n$，其中向量的每一个元素$X_i$是一个特征。

而所谓的学习，其本质就是找到特征与标签之间的映射(mapping)关系。所以说分类预测模型是求取一个从输入向量(特征)$\boldsymbol{X}$到离散的输出变量(标签)y之间的映射函数$f(x)$。这样，当有特征而无标签的未知数据输入时，可以通过映射函数预测未知数据的标签。

简单地说，分类预测的过程就是按照某种标准给对象贴标签，再根据标签区分归类。类别是事先定义好的。例如，在CTR(点击率预测)中，对于一个特定商品，某位用户可以根据过往的点击商品等信息被归为两类“会点击”和“不会点击”；类似地，房屋贷款人可以根据以往还款经历等信息被归为两类“会拖欠贷款”和“不会拖欠贷款”；一个文本邮件可以被归为“垃圾邮件”和“非垃圾邮件”两类。

5.1.1 分类与聚类的区别

分类与聚类是两个容易混淆的概念，实际上，它们是完全不同的概念。从它们的训练集上可以发现分类与聚类的明显区别。图5-1是分类训练集和聚类训练集的对比，该数据集表示豌豆种子在不同环境下能否发芽的情况。

从图5-1中可见，分类的训练集包含特征属性和类别属性，其中特征属性是一个向量$\boldsymbol{X}$，而类别属性是一个离散值变量y(该变量可取值个数为2，表示2分类问题，若可取值

个数大于 2，则表示多分类问题），因此分类的训练集可表示为（X，y），而聚类的训练集只有特征属性，聚类的训练集可表示为（X）。

分类的训练集

形状	颜色	大小	土壤	是否发芽
圆形	灰色	饱满	酸性	否
圆形	白色	皱缩	碱性	是
皱形	白色	饱满	碱性	否
圆形	青色	饱满	酸性	是
圆形	白色	皱缩	酸性	是
皱形	灰色	皱缩	酸性	否
圆形	白色	饱满	碱性	否

特征属性　　类别属性

聚类的训练集

形状	颜色	大小	土壤
圆形	灰色	饱满	酸性
圆形	白色	皱缩	碱性
皱形	白色	饱满	碱性
圆形	青色	饱满	酸性
圆形	白色	皱缩	酸性
皱形	灰色	皱缩	酸性
圆形	白色	饱满	碱性

特征属性

图 5-1　分类训练集和聚类训练集的对比

实际上，分类对应机器学习中的监督学习，而聚类是一种无监督学习。

另外，监督学习又可分为产生式模型（generative model）和判别式模型（discriminative model）两种。机器学习的任务是从特征属性 X 预测标签 y，即求条件概率 $P(y|X)$。对于判别式模型来说，对未知类别的样本 X，根据 $P(y|X)$可以直接求得标签 y，即可以直接判别出来。线性回归模型和支持向量机都属于判别式模型。而产生式模型需要求 $P(y,X)$，对于未知类别的样本 X，要求出 X 与不同类别之间的联合概率分布，然后比较大小。朴素贝叶斯模型、隐马尔可夫模型（hidden markov model，HMM）等都属于产生式模型。这两种模型暂时不需要读者理解，当把本书内容学完之后再仔细体会两种模型的区别。

5.1.2　分类的步骤

分类的目的是构造一个分类函数或分类模型（分类器），该模型能把一些新的未知类别的数据映射到某一个给定类别。对一个样本集进行分类，大致有 4 个步骤。

（1）将样本转化为等维的数据特征（特征提取）。这一步要求所有样本必须具有相同数量的特征，并兼顾特征的全面性和独立性。

例如，要根据动物的一些特征预测某种动物属于鸟类，还是哺乳动物。首先收集一些已知类别的动物样本，将样本转化为等维的数据特征，如表 5-1 所示。

（2）选择与类别相关的特征（特征选择），在表 5-2 中，粗体字的特征表示与类别非常相关，其他的特征表示与类别完全无关。那么，可以只保留粗体字的特征，而将其他特征（是否有毛）删除。

表 5-1 样本转化为等维的数据特征

动物种类	体型	翅膀数量	脚的只数	是否产蛋	是否有毛	类 别
狗	中	0	4	否	是	哺乳动物
猪	大	0	4	否	是	哺乳动物
牛	大	0	4	否	是	哺乳动物
麻雀	小	2	2	是	是	鸟类
天鹅	中	2	2	是	是	鸟类
大雁	中	2	2	是	是	鸟类

表 5-2 选择与类别相关的特征

动物种类	**体型**	**翅膀数量**	**脚的只数**	**是否产蛋**	是否有毛	类 别
狗	**中**	**0**	**4**	**否**	是	哺乳动物
猪	**大**	**0**	**4**	**否**	是	哺乳动物
牛	**大**	**0**	**4**	**否**	是	哺乳动物
麻雀	**小**	**2**	**2**	**是**	是	鸟类
天鹅	**中**	**2**	**2**	**是**	是	鸟类
大雁	**中**	**2**	**2**	**是**	是	鸟类

(3) 建立分类模型或分类器(分类)。

分类器通常可以看作一个函数,它把特征向量(feature vector)映射到类的空间上,该函数可表示如下:

$$f(x_{i_1},x_{i_2},x_{i_3},\cdots,x_{i_n})\rightarrow y_i$$

在本例中,特征向量 $\boldsymbol{X}$ 是(体型,翅膀数量,脚的只数,是否产蛋),类别属性 y 是{哺乳动物,鸟类}。

(4) 用建立的分类模型预测未知类别样本的所属类别。

例如,要对表 5-3 中的新发现物种:动物 A(大、0、2、是)和动物 B(中、2、2、否)进行分类,则只要将特征向量作为分类函数的输入,即可得到类别值。

表 5-3 预测未知类别样本的所属类别

动 物 种 类	体型	翅 膀 数 量	脚 的 只 数	是 否 产 蛋	类 别
动物 A	大	0	2	是	?
动物 B	中	2	2	否	?

(5) 分类模型预测结果的评估。

对于分类模型来说,必须评估模型的泛化能力(generalization ability)。所谓泛化能力,是指机器学习模型对新样本的适应能力,也就是说,分类模型对新样本的预测结果越准确,该分类模型的泛化能力越好。分类模型评估的具体方法和指标见 5.1.3 节。

总结，分类过程包括以下两个阶段。

（1）模型训练阶段：这一阶段使用训练集构建一个分类模型（或规则），如图 5-2 所示。

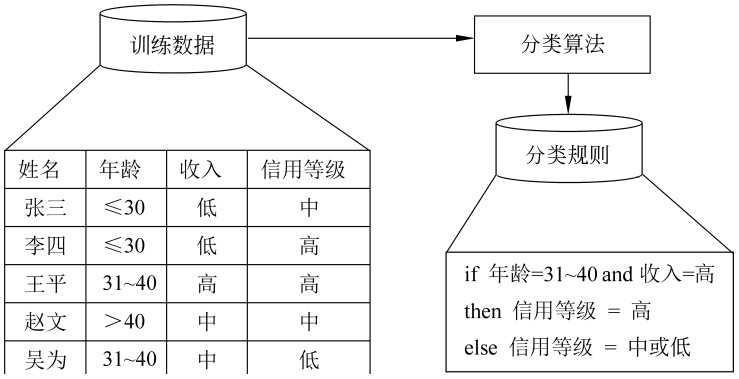

图 5-2　分类模型的训练阶段

（2）模型使用阶段：使用模型对测试数据或类别未知的新数据进行分类，如图 5-3 所示。对测试数据进行分类是为了评估分类模型的准确率，对新数据进行分类是为了预测新数据的类别。

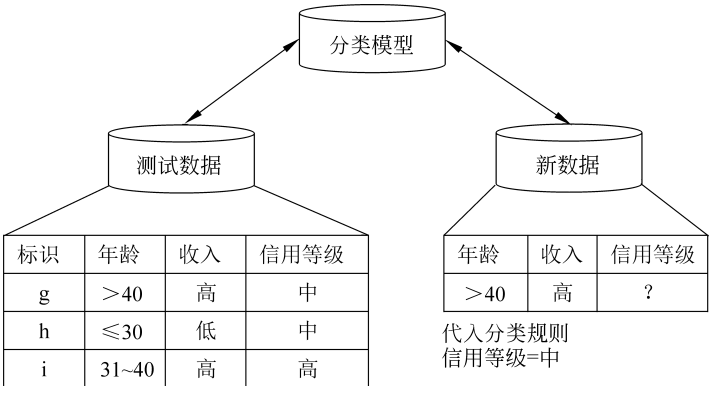

图 5-3　分类模型的使用阶段

5.1.3　分类模型预测结果的评估

在用分类模型预测测试集样本的所属类别之后，需要对分类模型的准确率进行评估，以判断该分类模型的准确率是否能够满足应用的需要。假设在表 5-3 中，动物 A、动物 B 的预测类别是{鸟类、哺乳动物}，而通过分类模型预测的结果是{鸟类、鸟类}，则称该模型的预测准确率(accuracy)为 50%。预测准确率高，也称为模型的泛化能力好。

但分类模型的预测结果不能单纯用准确率进行评估。为了有效地判断一个预测模型的性能表现，需要通过比较预测值和真实值，计算精确率、召回率、F1 值和 Cohen's Kappa

系数等指标衡量。常规分类模型的评估方法如表 5-4 所示。

表 5-4 常规分类模型的评估方法

方 法 名 称	最 优 值	Sklearn()函数
Precision(精确率)	1.0	metrics.precision_score
Recall(召回率)	1.0	metrics.recall_score
F1 值	1.0	metrics.f1_score
Cohen's Kappa 系数	1.0	metrics.cohen_kappa_score
ROC 曲线	最靠近 y 轴	metrics.roc_score

下面对分类模型的评估方法的含义进行介绍。以一个二分类问题为例,样本有正负两个类别。

那么,模型的预测结果和真实标签的组合就有 TP、FP、FN、TN 4 种,如表 5-5 所示。

表 5-5 预测结果和真实标签的组合

真 实 值	预 测 值	
	Positive	Negative
Positive	True Positive(TP)	False Negative(FN)
Negative	False Positive(FP)	True Negative(TN)

其中,TP 表示实际为正样本,预测值也是正样本(真阳性);FN 表示实际为正样本,预测值为负样本(假阴性);FP 表示实际为负样本,预测值为正样本(假阳性);TN 表示实际为负样本,预测值为负样本(真阴性)。

可见,TP 和 TN 都是预测正确的情况,因此,预测的准确率就可定义为

$$\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

而预测的精确率表示预测为正的样本中有多少是真正的正样本,精确率可定义为

$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP})$$

召回率表示样本中的正例有多少被预测正确了,召回率定义为

$$\text{Recall} = \text{TP}/(\text{TP} + \text{FN})$$

对于机器学习模型来说,当然希望 Precision 和 Recall 这两者都保持较高的水准,但事实上这两者在很多时候是不可兼得的。为此,提出了 F_1 分数(F1 Score)的概念,它同时兼顾了分类模型的精确率和召回率。F_1 分数可以看作模型精确率和召回率的一种加权平均,它的最大值是 1,最小值是 0。其定义如下。

$$F_1 = 2 \times \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

5.1.4 Sklearn 库的常用分类算法

在机器学习领域,分类算法很多,其原理千差万别。有基于样本距离的 k-近邻算法,

有基于贝叶斯定理的朴素贝叶斯算法，有基于信息熵的决策树算法，有基于 bagging 的随机森林算法等。

Sklearn 提供了几乎所有目前常用的分类算法，并且分别存在于不同的模块中，如表 5-6 所示。这与聚类算法不同，Sklearn 中所有的聚类算法都位于 cluster 模块中。

表 5-6　Sklearn 库中常用的分类算法

类　　名	所 在 模 块	算 法 名 称
neighbors	KNeighborsClassifier	*k*-近邻分类
GaussianNB	naive_bayes	高斯朴素贝叶斯分类
DecisionTreeClassifier	tree	决策树分类
RandomForestClassifier	ensemble	随机森林分类
LogisticRegression	linear_model	逻辑回归
SVC	svm	支持向量机

5.2　*k*-近邻分类算法

物以类聚，人以群分。判别一个人是什么品质的人，常常可以通过他/她周围的朋友判断，所谓观其友，而识其人。*k*-近邻（k-Nearest Neighbor，KNN）分类算法就是一种以聚类的思想做分类的分类算法，它是最简单的机器学习算法之一。该算法最初由 Cover 和 Hart 于 1968 年提出，它根据距离函数计算待分类样本 X 和每个训练样本的距离（作为相似度），首先选择与待分类样本距离最小的 k 个样本作为 X 的 k 个最近邻，然后以 X 的 k 个最近邻中的大多数样本所属的类别作为 X 的类别。

5.2.1　*k*-近邻算法原理和实例

所谓 *k*-近邻算法，即给定一个训练数据集，对新的输入实例，在训练数据集中找到与该实例最邻近的 k 个实例（也就是所谓的 k 个邻居），若这 k 个实例中多数属于某个类，就把该输入实例分类到这个类中。

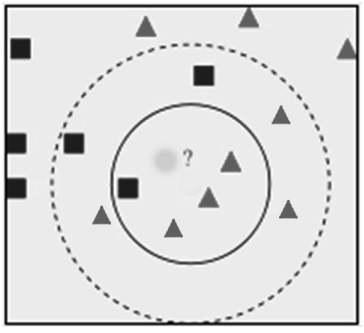

图 5-4　KNN 算法示意图

如图 5-4 所示，有两类不同的样本数据（L_1 和 L_2），L_1 用小正方形表示，L_2 用小三角形表示，图中间的圆表示的是未知类别的待分类的数据 X。现在，要对 X 进行分类，判断它属于 L_1 还是属于 L_2。

k-近邻分类的过程是：先主观设置 k 的值，设 $k=4$，然后寻找与 X 距离最近的 4 个点，从图 5-4 中可发现 X 的 4 个近邻点中有 3 个属于 L_2 类，有 1 个属于 L_1 类，可见 4 个最近邻点大多数属于 L_2 类，从而可判断 X 的类别是 L_2。

1. k-近邻算法的步骤及举例

k-近邻算法的实现，大致包括以下 3 个步骤。

(1) 算距离：给定测试对象，计算该对象与训练集中每个对象的距离。

(2) 找邻居：圈定距离最近的 k 个训练对象，作为测试对象的近邻。

(3) 做分类：这 k 个近邻对象大多数归属的类别，作为测试对象的类别。

k-近邻算法的编程步骤如下。

(1) 初始化距离为最大值。

(2) 计算未知样本和每个训练样本的距离 dist。

(3) 得到目前 k 个最近邻样本中的最大距离 maxdist。

(4) 如果 dist 小于 maxdist，则将该训练样本作为 k-最近邻样本。

(5) 重复步骤(2)～步骤(4)，直到未知样本和所有训练样本的距离都算完。

(6) 统计 k 个最近邻样本中每个类别出现的次数。

(7) 选择出现频率最高的类别作为未知样本的类别。

【例 5-1】 表 5-7 是一个二手房分类的训练集，对于二手房的新样本 $T=\{18,8\}$，试用 KNN 算法预测其所属类型。

表 5-7 二手房分类的训练集

序 号	房 龄	与市中心距离	类 型
A	2	4	L_1
B	4	3	L_2
C	10	6	L_3
D	12	9	L_2
E	3	11	L_3
F	20	7	L_2
G	22	5	L_2
H	21	10	L_1
I	11	2	L_3
J	24	1	L_1

解：本例采用欧氏距离作为距离度量方法，且设 $k=4$。

首先计算新样本 T 与训练集中所有样本的距离。例如 T 与 A 的距离为 $d(T,A)=\sqrt{(18-2)^2+(8-4)^2}=16.5$，$T$ 与所有样本的距离如表 5-8 所示。

表 5-8 新样本 T 与训练集中所有样本的距离

序 号	A	B	C	D	E	F	G	H	I	J
与样本 T 的距离	16.5	14.9	8.3	**6.1**	15.3	**2.2**	**5**	**3.6**	9.2	9.2

然后，找距离最近的 k 个邻居，分别是$\{F,H,G,D\}$，这 4 个邻居对应的类别分别是$\{L_2,L_1,L_2,L_2\}$，所以新样本 T 所属的类别是 L_2。

下面绘制如图 5-5 所示的散点图，用来直观地观察 T 所属类别是否正确。从图 5-5 中可见，新样本 T 确实与 F、H、G、D 这 4 个点的距离最近。

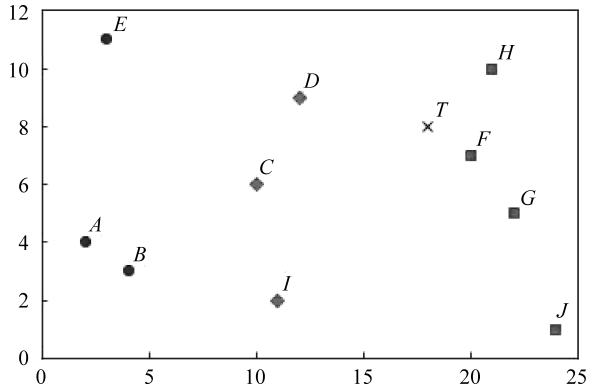

图 5-5　例 5-1 中数据的散点图

2. k-近邻算法的优缺点

k-近邻算法的优点：

(1) 算法思路简单，易于实现，对异常值不敏感，无数据输入假定。

(2) 当有新样本要加入训练集中时，无须重新训练(即重新训练的代价低)。

(3) 计算时间和空间线性于训练集的规模，对某些问题而言这是可行的。

其缺点及适用数据范围如下。

(1) 分类速度慢，该算法的时间复杂度为 $O(m*n)$。

(2) 各属性的权重值相同时，可能影响准确率。

(3) 样本库容量依赖性较强，当样本容量太小时，会影响预测准确率。

(4) k 值不好确定。

3. k-近邻算法常见问题及解决方法

在实际应用中，k-近邻算法可能会遇到以下 3 个需要解决的问题。

1) 样本不平衡时对算法的影响

如图 5-6 所示，k-近邻算法在分类时有个很大的不足之处是，当样本不平衡时，即一个类的样本容量很大，而其他类的样本容量很小时，很有可能导致当输入一个未知样本时，该样本的 k 个邻居中大数量类的样本占多数。例如，Y 从图 5-6 中看应属于 ω_1 类，但应用 KNN 算法会将其错误划分到 ω_2 类中。

为此，可以采用对近邻点赋权值的方法改进。和该样本距离小的邻居权值大，和该样本距离大的邻居权值则相对较小。由此，将距离远近的因素也考虑在内，避免因某个类别样本的容量过大而导致误判的情况。

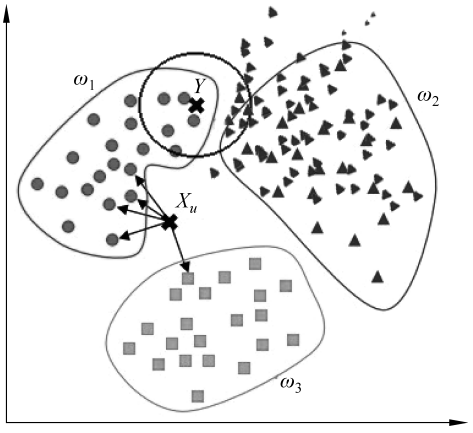

图 5-6 样本不平衡时 KNN 算法预测效果

2) k 值取值对算法的影响

在 k-近邻算法中，k 值是主观设定的。如图 5-7 所示，当增大 k 值时，一般分类错误率会先降低，因为有周围更多的样本可以借鉴。但是，当 k 值更大的时候，错误率会更高。这也很容易理解，比如样本集中一共就 35 个样本，当把 k 值增大到 35 时，k-近邻算法基本上就没意义了。

要选出最优的 k 值，思路是：分别尝试不同 k 值下分类模型的准确率，这可以使用 Sklearn 中的交叉验证方法或网格搜索方法。

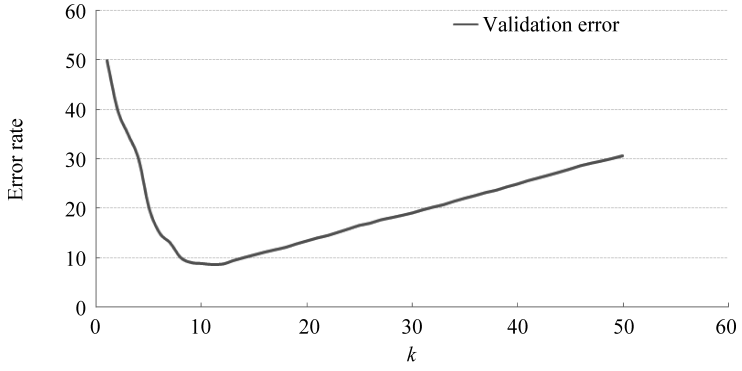

图 5-7 k 值与分类错误率的关系

3) 如何快速找到待测点的 k 个最近邻

最基本的 k-近邻算法需要计算待测点与训练集中所有数据点之间的距离，如果训练集中的点(样本)很多，则这个操作会很耗时，然后还需要对所有的这些距离进行排序，才能找到距离最小的 k 个数据点，这个排序操作也比较耗时。

为了解决上述问题，人们提出了 kd 树(k dimensional tree)算法，kd 树可以快速找到与待测点最邻近的 k 个训练点，而不需要再计算待测点与训练集中的每个数据点的距离。

kd 树算法类似于“二分查找”。kd 树是二叉树的一种，是对 k 维空间的一种分割，如图 5-8 所示。构造 kd 树相当于不断地用垂直于坐标轴的超平面将 k 维空间切分，构成一系列的 k 维超矩形区域。kd 树的每个结点对应一个 k 维超矩形区域。利用 kd 树可以省去对大部分数据点的搜索，从而减少搜索的计算量。kd 树的具体原理可查看二维码中的视频。

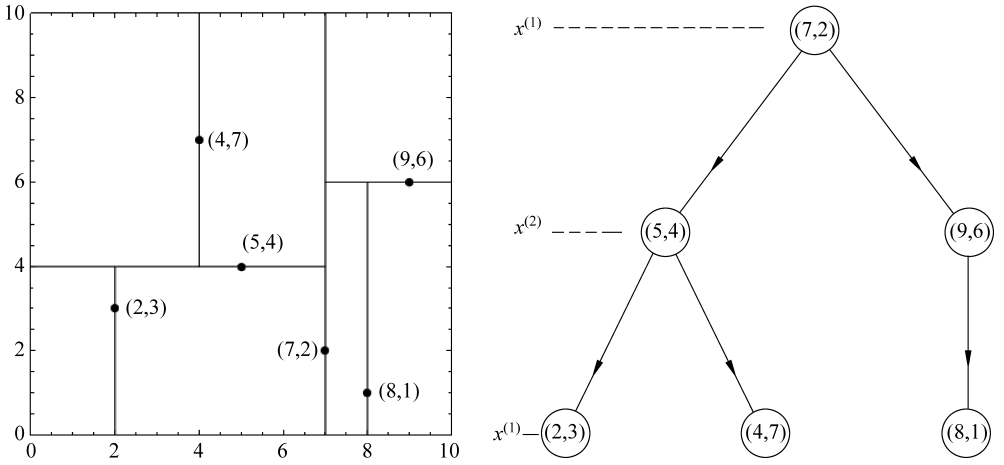

图 5-8　kd 树示意图

5.2.2　Sklearn 中分类算法的编程步骤

在 Sklearn 中，使用分类算法的编程步骤大致如下。

(1) 导入相应的机器学习及数据可视化模块，代码如下。

```
import matplotlib as mpl
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier          #导入 KNN
```

(2) 读取数据到 numpy 数组中，并将数据集划分为特征属性集 X 和标签集 y，代码如下。

```
X1,y1=[],[]
fr=open('D: \\knn.txt')
for line in fr.readlines():
   lineArr=line.strip().split()
   X1.append([int(lineArr[0]),int(lineArr[1])])
   y1.append(int(lineArr[2]))
X=np.array(X1)          #转换成 numpy 数组,X 为特征属性集
y=np.array(y1)          #y 为标签集
```