第 11 章 数据处理和挖掘

我们正步入一个数据或许比软件更重要的新时代。

We're entering a new world in which data may be more important than software.

——蒂姆·奥莱利(Tim O'Reilly)

O'Reilly 媒体公司的创始人兼 CEO、Open Source 和 Web 2.0 概念的提出者

学习目标

1. 了解数据挖掘的基本概念；
2. 掌握数据挖掘的基本方法；
3. 运用第三方库实现数据挖掘的聚类、分类和回归方法；
4. 了解机器学习的基本概念；
5. 掌握一种聚类或分类方法。

数据挖掘，又译为资料探勘、数据采矿。数据挖掘就是在大量数据中寻找有意义、有价值信息的过程。当别人空谈故事时，我们要学会用数据说话。在现代社会中，与其求助于个人的信息知识储备，不如借助网络的海量信息筛选来完成特定的任务，这个特定的任务对象并不一定要是枯燥的表格，它可以是一颗行星的轨道，可以是云雨的变化，或者是，一朵花？

本章将讲述如何使用 K 均值方法根据花瓣形状将花朵们分门别类。

11.1 问题概述

要点：数据挖掘是人类从数据中获得信息的手段。

——你是穷人还是富人？你学识渊博吗？

——没有横向比较，我不知道。

人们总喜欢对遇到的事物进行分类，一种直观的分类方法是利用严格的准则，如考试 60 分及格、盗窃侵权违法等。然而，更多的事物很难使用客观准则进行分类，例如，高考的录取分数每年要根据考生成绩和高校录取人数确定，最后确定的录取线是数据间相对比较的结果。计算机科学中的数据挖掘技术可以帮助人们更好地理解数据之间的相互关系及数据背后的奥秘，更有效地使用数据，通过数据形成有价值的判断标准，甚至“知识”，从而预测未来走势或支撑各种决策。数据挖掘是计算机科学的一个分支，有比较深厚的理论基础和技术框架。本书仅结合 Python 语言介绍一些入门知识。

理解和处理数据依次有 4 个层次：数据表示和清洗、数据统计和建模、数据挖掘和分析、知识形成和决策支持。数据表示和清洗指对获取数据进行归一化处理，消除无效内容，建立统一的表示形式，这是进一步数据分析的基础，见第 7.4 到 7.7 节。数据统计和建模通过数学方法对数据进行基本的量化统计，获取数据规模和形态的外部特征，见第 3.3 节、第 6.3 节和第 10 章。数据挖掘和分析从数据之间关联关系和内容判定角度发现数据表现的内在规律性内容，这是本章的主要内容。分析数据的最高境界是从数据中获得知识，辅助决策，这需要将数据与所处领域关联，至今仍然是数据领域科学家在研究的重要课题。

数据处理挖掘主要有三种方法：分类、聚类和预测。

分类首先使用有标签数据进行训练，然后建立系统对未分类数据进行标签判定。标签数据是一种经过判定的数据，例如，给定一组与飞机有关的图片，每张图片都经过人工判断，确定与飞机有关。未分类数据，也称为无标签数据，指未经过人工判断的数据。分类方法通过学习经过判定的数据形成分类器，可以对未分类数据进

行判定。

聚类主要针对现实中缺少有效标签数据或者标签数据规模不大的情况，聚类通过对无标签数据属性特征的学习，发现数据之间的“相似性”，实现数据聚合。例如，给定一组图片，其中包括飞机的图片和汽车的图片，在不需要人工判断的前提下，聚类方法能够将飞机的图片形成一组，汽车的图片形成另一组。尽管聚类可以通过特性将数据分成两组，但却不能解释图片内容是飞机还是汽车。

预测用于分析连续变量之间的相互依赖关系，预测变量未来的变化。预测是一个非常有魅力的词汇，通过过去预测未来是人类一直孜孜以求的目标。例如，分析股票市场过去一段历史时期的规律，判断近期或未来某只股票的涨跌情况，如果这个判断准确将直接赚取大量财富，使用数据挖掘方法辅助股票交易决策形成了“量化交易”这个领域。

拓展：量化交易

量化交易是金融和计算机领域的交叉应用，指以股市历史交易数据为基础，采用先进数学模型替代人为主观投资判断的金融交易方式。量化交易能够利用计算机快速计算能力和骨干网络带宽从庞大历史数据中发现收益大风险低的投资组合、捕捉瞬息万变的投资机会。量化交易主要采用数据挖掘算法。

掌握分类、聚类和预测三种方法是理解和利用数据的重要进阶能力，然而，对非专业人士来说，这些方法似乎过于高深，掌握其中原理和算法十分困难。事实的确如此，较高的理论门槛限制了数据挖掘在其他领域的应用程度和范围。

Python 语言的普及为数据挖掘更广泛应用带来了新的契机，专业人士用 Python 语言构建了一批非常优秀的数据挖掘函数库，对于非专业人士应用数据挖掘技术不再需要理解深奥的数学模型，只需结合拟解决的问题选择合适方法即可。Python 语言的模块编程思想极大简化了数据挖掘应用的门槛，给深入数据处理和挖掘提供了有力支撑。

本章将介绍一个 Python 语言中非常优秀的数据挖掘第三方函数库——sklearn。

与其他第三方库一样，采用 pip 指令安装 sklearn 库，如果在 Python 2 和 Python 3 并存的系统中，采用 pip3 指令，如下。

```
:\>pip install sklearn   # 或者 pip3 install sklearn
```

本书之前讲解了不少 Python 库，读者对使用库应该有了较好的理解，因此，本章将先给出使用 sklearn 库的简单使用实例，见第 11.3 节，再详细讲解这个库的用法，见第 11.4 节。鉴于数据挖掘本身很有趣，第 11.2 节对这个概念进行了极简介绍，便于读者了解一些背景知识。

思考与练习：

[E11.1]思考如何在一个群体里发现兴趣小组或小团体？

[E11.2]思考数据挖掘可能的应用都有哪些？

11.2 极简数据挖掘

要点：数据挖掘从简单的数据模型分析已经逐步向人工智能方向发展，这种基于数据分析而非逻辑推理的智能方法将伴随计算机性能的提升应用到更加广阔的领域。

人类行为和自然运动无时无刻不在产生数据，在没有计算机的时代，即使统计很小量数据都需要花费巨大的人力和物力成本。因此在人类漫长且几乎全部的历史中，从简单现象抽象规律、直接研究符号逻辑和推理成为了主流，这些物理的、数学的方法形成了严密的思维体系，指导探索人类和自然世界。这个阶段对现象的认识采用“猜想证明”的思维模式，即首先猜想规律，再通过逻辑证明。

计算机快速的数据处理能力让人类可以从数据角度思考世界。对于一个问题的思考不再通过猜想证明方式去探求因果关系，而是采用“数据关联”模式通过获得数据并发现数据之间的关联性，进而理解问题，这种思考模式产生了数据挖掘。

数据挖掘的概念伴随着计算机的进步而不断更新，不同发展阶段数据挖掘的使命、任务和方法都不相同，产生了丰富的数据挖掘算法和概念。数据挖掘技术应用广泛，例如，银行通过对客户数据进行分类构造一个分类模型，进而评估银行贷款的风险；医生通过病人的电子病历信息预测未来的患病概率；根据社交网络行为和好友关系进行聚类，发现不同的兴趣小组。

在电子邮件刚刚普及的年代，垃圾邮件困扰着很多用户，甚至有的用户每天收到上百封垃圾邮件而导致邮箱无法正常使用。邮件服务商为了提升服务水平改善用户体验，纷纷采用数据挖掘方法对垃圾邮件进行自动识别和拦截，帮助用户避免垃圾邮件的侵扰。数据挖掘方法通过对邮件发件地址和邮件内容的分析，部分借助工程师人工标记或用户对垃圾邮件的反馈，形成了规模庞大的垃圾邮件数据库。针对这个数据库，数据挖掘方法进一步找到了常见垃圾邮件的模型，并不断识别垃圾邮件最常用的词语和主题，最终形成了识别常见垃圾邮件的全自动分类器。当用户收到一封新的邮件时，分类器程序会扫描邮件内容，根据判断结果确定是否对邮件进行拦截。垃圾邮件分类器准确且高效，有效证明了数据挖掘技术的实用性。

互联网技术快速发展初期，各种网站不断涌现，网页广告也随之出现并广泛应用。最开始，企业往往不加区分地投放广告，导致大量预算浪费却达不到宣传目的。以谷歌为代表的互联网公司发现了这个重要问题，推出精准的定点广告投放服务，帮助企业将广告投放到最合适的网站上，进而最大化宣传效果。定点广告投放的关键是精准的网站分类，对于人来说，区分一个科技网站和文学网站十分容易，然而让机器识别却需要数据挖掘方法。此外，活跃网站数量一般在几百万规模，人工分类效率无法满足需求。在这个时期，数据挖掘算法通过分析网站内容、超链接关系、用户浏览量和平均浏览时间等指标，建立网站的分类体系，为企业挑选最符合其广告的发布平台。例如，销售电脑配件公司的广告将被自动定点投放在科技新闻或电脑爱好者论坛类网站，销售女性服装公司的广告将新投放在时尚类或化妆品类网站。数据挖掘极大改善了企业营销行为，也避免了用户过度浏览无效广告的困扰，可谓一举多得。

互联网进一步发展到了电商时代，用户渐渐习惯了网上购物也学会了无视网页

上被投放的广告，企业急需一种新方法来推广产品刺激用户购买。个性化推荐系统顺应这股潮流，成为数据挖掘发展的新动力。实际生活中，人们对事物的描述通常会基于不同属性的量化表示，如汽车的耗油量和颜色等。这种量化指标可以是离散的标签，也可以是连续的数值。对于电商时代的用户，也有如名字、性别、年龄、身高体重、兴趣好爱等属性。个性化推荐系统充分利用用户属性构建了虚拟用户形象，通过挖掘分析用户历史购物数据，获得用户购物习惯并推荐给用户可能喜欢的商品，进而刺激用户购买。如今，每个购物网站都会有一个“猜你喜欢”区域，为用户个性化推荐商品。这些商品可能是通过关联规则筛选出的搭配组合，也可能是根据历史购物信息预测的结果。数据挖掘方法按照不同的用户属性或属性组合进行用户分类，为每个用户指定个性化的营销策略。

数据挖掘在上述历史时期发展中形成了一批算法，常用算法例如：决策树、K均值（K-means）、逻辑回归（Logistic Regression）、最近邻（KNN，K-NearestNeighbor）、支持向量机（SVM）、贝叶斯法和人工神经网络等。其中，KNN 算法是对有标签数据进行分类的常用方法；K 均值算法则是将无标签数据聚合成 K 个类的常用聚类算法；逻辑回归针对连续数据进行变量关系分析，它是一个简单易行的预测方法。

本书撰写之际，基于数据挖掘的深度学习技术正热火朝天地影响着人工智能领域的发展。谷歌公司的 AlphaGo 计算机及算法以 4:1 的大比分优势战胜了世界围棋冠军、职业九段选手李世石，让人瞠目结舌，人工智能威胁论也因此甚嚣尘上，AlphaGo 所采用的深度学习技术也极罕见的发表在《自然》杂志上。AlphaGo 采用的深度学习技术并不是崭新的技术，它是由计算机科学传统的人工神经网络技术演变而来，该技术早期主要用于发现数据的分布式特征。甚至，人工神经网络计算模型因为其巨大的计算需求而被诟病，直到近些年，计算机硬件速度高速发展，GPU 和专用硬件为计算神经网络模型提供了保障，促使深度学习快速发展。

数据挖掘从简单的数据模型分析已经逐步向人工智能方向发展，这种有料的智能技术已经广泛应用在计算机视觉、语音识别、自然语言处理等领域。可以预见，这种基于数据分析而非逻辑推理的智能方法将伴随计算机性能的提升应用到更加广阔的领域，“人工智能阶段”还会远吗？

拓展：强人工智能

强人工智能指计算机只要运行适当程序就能够产生思维，进而快速进化超越人类思维。与传统人工智能根据历史规律学习并表现出明显“智能”不同，强人工智能认为计算机不仅是一种工具,更是有可能产生思维的计算载体,它能够真正推理、解决问题、有知觉和自我意识。如果计算机真能够达到强人工智能阶段，它将不再是工具，而是一个新的物种。

思考与练习：

[E11.3]请用自己的语言阐述对数据挖掘的思考。

[E11.4]常用的无标签数据聚类算法是什么？有标签数据的分类算法呢？

[E11.5]列出 5 个数据挖掘方法的应用。

11.3 实例 22：物以类聚、花以瓣儿分

要点：这是一个利用 sklearn 库采用聚类方法辨识鸢尾花种类的实例。

春天，走在开满各式艳丽花朵的植物园里，偶然瞥见一种紫色的花朵，它看上去陌生但美丽，花坛的标签上写着“鸢尾花”（鸢，音 yuān，英 iris）三个字。对于这个新奇的发现，兴奋之余写个小心情发到朋友圈炫耀一下。也许会有朋友问这样的问题：“看见的是哪种鸢尾花？山鸢尾（Setosa），杂色鸢尾（Versicolour）还是维吉尼亚鸢尾（Virginica）？”。标签上只写了花名却没说明品种，只好干脆拍张照片再发到朋友圈里。不久有了新的反馈，原来这朵紫色的小花是维吉尼亚鸢尾。此时，也只剩下心情去赞叹这位朋友的渊博知识和凭借照片辨识品种的能力了。其实，不同品种的鸢尾花有不同的花瓣形状，根据这个可以很方便分辨具体种类。网上有个著名的 IRIS 鸢尾花数据库，记录了这三个品种的花瓣数据。一起来数花瓣吧！

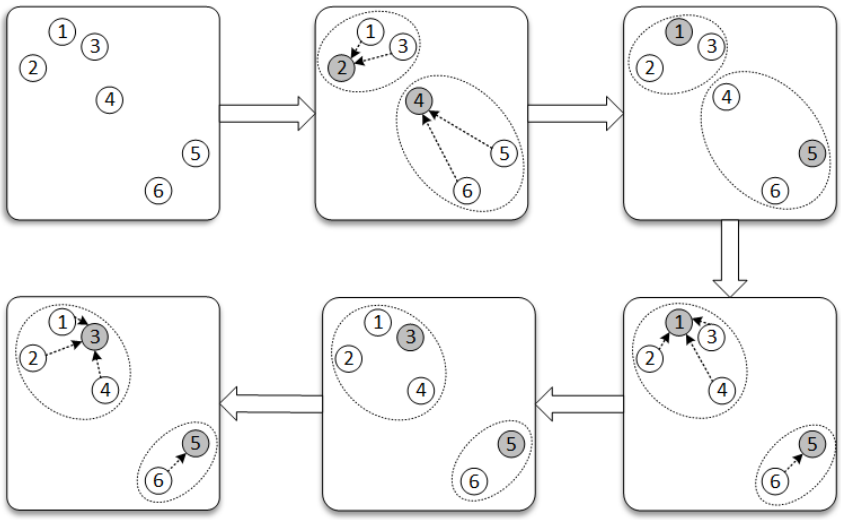
IRIS 数据集包含 150 条数据，每个数据源于一朵花，记录了该花的花萼长度、

花萼宽度、花瓣长度、花瓣宽度等 4 个属性。随后再由人工逐一判断花朵类型，将 150 条数据标定成鸢尾花卉的三个具体种类：Setosa、Versicolour 和 Virginica，其中，每种类型恰好 50 个数据。简单说，IRIS 数据集包含来源于鸢尾花三个类别的各 50 个数据，共 150 条数据。由于鸢尾花的 Setosa、Versicolour 和 Virginica 三个子类别在进化中出现了不同外表，可以通过花瓣形状进行分辨，因此，IRIS 数据集成为了数据挖掘和数据分类中十分常用的测试集和训练集。

物以类聚，花以瓣儿分，这节主要讲解聚类方法。聚类是一种主要的数据挖掘与机器学习方法，而 Kmeans（K 均值）是聚类方法中使用最广泛的方法之一。

K 均值是一种非常典型的基于划分的聚类算法，核心思想是把一堆离散的点根据距离分为 K 组。算法过程如下：首先随机选择 K 个对象，每个对象初始地代表一个簇的平均值或者中心。对于剩余的每个对象，根据其到各个簇中心的距离，把它们分给距离最小的簇中心，然后重新计算每个簇的平均值。

这里以 6 个点聚类的简单实例说明 K 均值方法，聚类过程如图 11.1 所示。



[图 11.1: K 均值方法的简单实例]

首先确定目标，即将 6 个点聚类成两个簇（部分）。开始时，在 6 个点中随机选取 2 个中心点（例如 2 和 4），然后让其他点逐个归类到离它最近的中心点，形成第一次聚类，结果为两个簇(1，2，3)和(4，5，6)。接下来，重新计算中心点，选取当前簇的中心点，即 1 和 5，计算其它点到 1 和 5 的距离，再次形成两个簇，第二次

聚类结果为(1，2，3，4)和(5，6)。再次找两个簇的中心点，即 3 和 5，计算各点到中心点距离重新聚合后形成的两个簇，聚类结果仍然是(1，2，3，4)和(5，6)，聚类结束。K 均值算法的 IPO 描述如下：

输入：数据点、聚类目标 K

处理：

步骤 1：随机选取 K 个点作为初始中心

步骤 2：计算数据点到 K 个点距离，将数据点分成 K 个簇

步骤 3：根据簇计算 K 个簇中心点，当作新的簇中心

步骤 4：如果 K 个簇聚类有变化，重复步骤 2 和步骤 3；否则退出

输出：包含全部数据点的 K 个簇

在“物以类聚、花以瓣儿分”实例中，采用 IRIS 数据集的 150 个数据作为源数据，并且利用 sklearn 库中的 kmeans 函数作为聚类算法。由于已知鸢尾花有 3 个种类，所以，初始设定将 150 个数据聚为 3 类。

根据程序处理流程，这个聚类程序可以分成四个步骤：

（1）从 sklearn 库导入 IRIS 数据集，将其存储在 Numpy 的数据对象中；

（2）指定聚类个数和最大迭代次数；

（3）调用 sklearn 库中 kmeans 聚类函数，得到聚类结果；

（4）将聚类结果以图形方式展示出来。

本书之前章节介绍了很多 Python 标准库或第三方库，因此，这里假设读者熟悉库函数的使用。本实例拟先介绍 sklearn 库的一个应用实例，将在第 11.4 节全面介绍 sklearn 库的使用，由于 sklearn 库不能够完全采用 pip 安装，鉴于读者先浏览第 11.4 节中与 sklearn 库安装相关的内容。

第一步，从 sklearn 库中导入 datasets 类，然后调用它的 load_iris()方法读取 IRIS 数据集并赋给变量 iris。同时，将 iris 变量的 data 成员赋给 dataSet 变量。可以通过 print(dataSet)语句打印 IRIS 数据集内容。

```
from sklearn import datasets
iris = datasets.load_iris()
dataSet = iris.data
```

第二步，获取用户输入的聚类个数 k 以及最大迭代次数 iterNum。

第三步，调用 Sklearn 的 K 均值方法聚类数据集 dataSet，这两步代码如下。

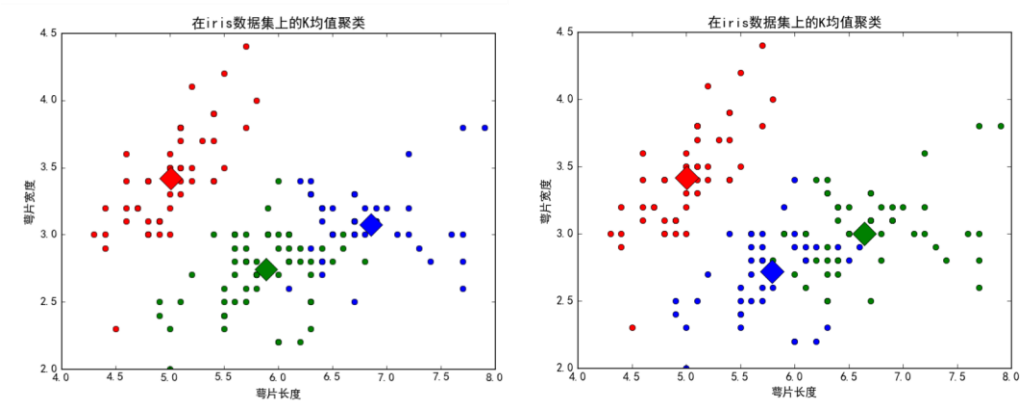
```
from sklearn.cluster import KMeans
k = eval(input("请输入 K 值： "))
iterNum = eval(input("请输入最大迭代次数： "))
model= KMeans(n_clusters=k, init='random', max_iter=iterNum)
model.fit(dataSet) #对数据集开始聚类
```

第四步，为了将聚类结果显示出来，定义显示函数 showCluster()，将聚类结果用不同颜色显示在坐标系中。该函数首先计算数组 dataSet 的行数和列数，并定义 5 种颜色，然后利用 for 循环根据类别绘制每个样本点，最后利用 for 循环把每个中心点以更大的尺寸绘制出来，该函数代码如下。

```
def showCluster(dataSet, k, centers, clusters):
    plt.figure(facecolor = 'white')
    num, nattr = dataSet.shape   #数据个数和属性个数
    mark = ['or','ob','og','om','oy'] #聚类颜色
    for i in range(num):
        plt.plot(dataSet[i,0],dataSet[i,1],mark[int(clusters[i])])
    mark = ['Dr', 'Db', 'Dg','Dm', 'Dy']
    for i in range(k): # 绘制 K 均值各簇中心点
        plt.plot(centers[i,0],centers[i,1],mark[i],markersize = 17)
```

```
    plt.title("在 iris 数据集上的 K 均值聚类")
    plt.xlabel("萼片长度")
    plt.ylabel("萼片宽度")
    plt.show()
```

将萼片长度作为横轴，萼片宽度作为纵轴，以二维方式将聚类结果展现出来，其中，每种颜色代表一种分类。由于 K 均值的初始簇中心点是随机选取的，有可能出现多次运行结果不一致的情况，图 11.2 是运行两次程序后的聚类结果。



[图 11.2: iris 数据集上的聚类结果]

实例 22 的全部代码如下。

实例代码 22.1　　e22.1ClusterIRIS.py

```
#e22.1ClusterIRIS.py
from sklearn.cluster import KMeans
from sklearn import datasets
import matplotlib.pyplot as plt
import matplotlib
import numpy as nps
matplotlib.rcParams['font.family']='SimHei'
matplotlib.rcParams['font.sans-serif'] = ['SimHei']
```

```
def showCluster(dataSet, k, centers, clusters):
    plt.figure(facecolor = 'white')
    num, nattr = dataSet.shape   #数据个数和属性个数
    mark = ['or','ob','og','om','oy'] #聚类颜色
    for i in range(num):
        plt.plot(dataSet[i,0], dataSet[i,1], mark[int(clusters[i])])
    mark = ['Dr', 'Db', 'Dg','Dm', 'Dy']
    for i in range(k): # 绘制 K 均值各簇中心点
        plt.plot(centers[i,0], centers[i,1], mark[i], markersize = 17)
    plt.title("IRIS 数据集的 K 均值聚类")
    plt.xlabel("萼片长度")
    plt.ylabel("萼片宽度")
    plt.show()
def main():
    iris = datasets.load_iris()
    dataSet = iris.data
    k = eval(input("请输入 K 值： "))
    iterNum = eval(input("请输入最大迭代次数： "))
    model= KMeans(n_clusters=k, init='random', max_iter=iterNum)
    model.fit(dataSet) #对数据集开始聚类
    showCluster(dataSet, k, model.cluster_centers_, model.labels_)
main()
```

拓展：物以类聚的度量

“物以类聚，人以群分”出自《战国策》，比喻同类的东西常被放在一起，志同道合的人相聚成群。实例 22 用鸢尾花说明了物以类聚现象及度量这个现象的方法。分析物以类聚现象首先需要建立度量标准，即明确哪些特性（属性）使得事物之间产生了关联，进一步分析关联的程度。鸢尾花的聚类利用了花萼长度、花萼宽度、花瓣长度、花瓣宽度等属性，在数据挖掘中，确定度量属性是聚类的重要前提。

思考与练习：

[E11.6]鸢尾花的花语是什么？

[E11.7]画出 K 均值算法的流程图。

[E11.8]在实例代码 22.1 中，若要增加聚类的个数需要修改哪个参数？传递给 kmeans 函数的参数 iterNum 表示什么意思？

11.4 模块 12：sklearn 库的使用

要点：sklearn 库是汇集数据挖掘算法的第三方函数库。

11.4.1 sklearn 库概述

sklearn 库是 Python 语言中最优秀的数据挖掘库，它以 numpy+mkl 和 scipy 库为基础。如果读者使用 Windows 操作系统，numpy+mkl 和 scipy 无法用 pip 工具直接安装，请采用 8.6 节介绍的文件安装方式安装这两个基础库。

首先，从 http://www.lfd.uci.edu/~gohlke/pythonlibs 网站下载如下两个文件到 D:\pycodes 目录：

numpy-1.11.2rc1+mkl-cp35-cp35m-win32.whl

scipy-0.18.1-cp35-cp35m-win32.whl

然后，启动命令行执行如下命令：

```
:\>pip install "D:\pycodes\ numpy-1.11.2rc1+mkl-cp35-cp35m-win32.whl"
:\>pip install scipy-0.18.1-cp35-cp35m-win32.whl
```

最后，通过 pip 安装 sklearn 库

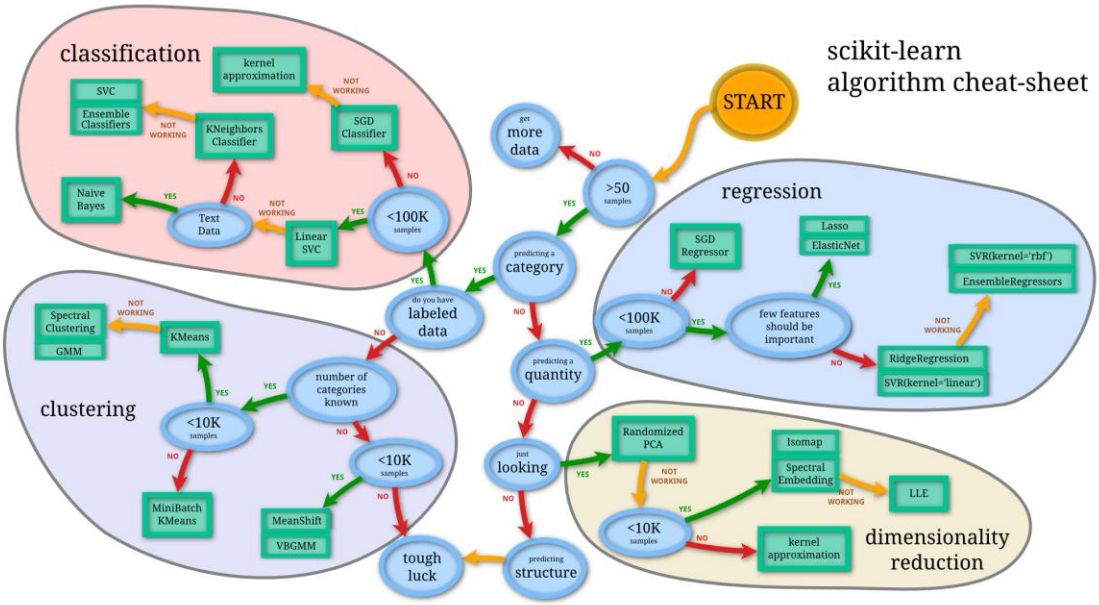
```
:\>pip install sklearn
```

数据挖掘算法按照是否需要训练可以分成两个大类：监督学习(Supervised learning)算法和非监督学习(Unsupervised learning)算法。监督学习算法首先需要对已有数据进行学习，进而对新样本或数据进行分类或预测，这类似人类学习知识和运用知识的过程。监督学习算法要求样本数据具有标签属性，即带有人为给出的判定，进而通过学习数据和判定之间关系，算法能够对新的数据开展分类或回归分析。无监督学习算法则不需要对已有数据进行学习，而是直接在数据中寻找关联，挖掘样本中隐藏的规律与性质，比如实例 21 对数据的聚类分析等。

sklearn 库提供了数据挖掘中涉及的数据预处理、监督学习、无监督学习、模型选择和评估等系列方法，包含众多子库或模块，例如 datasets 子库。sklearn 库常见的引用方式如下：

```
>>> from sklearn import <模块名>
```

sklearn 库主要提供聚类、分类和回归等三个主要功能，图 11.3 是 sklearn 库官方网站给出的使用该库的一个引导图，供有需求读者使用。本章主要围绕聚类、分类和回归介绍 sklearn 库的一些基本使用。



[图 11.3：sklearn 库使用引导图［此处有参考文献引用 x]]

拓展：机器学习

机器学习和数据挖掘是经常一起提及的两个相关词语。机器学习是数据挖掘的一种重要工具。数据挖掘不仅要研究、拓展、应用一些机器学习方法，还要通过许多非机器学习技术来解决例如数据仓储等更为实践的问题。机器学习应用广泛，不仅可以用在数据挖掘领域，还可以应用到与数据挖掘不相关的其他领域，例如增强学习与自动控制等。总体来说，数据挖掘是从应用目的角度定义的名词，而机器学习则是从方法过程角度定义的名词。

sklearn 库对所提供得各类算法进行了较好的封装，几乎所有数据挖掘算法都可以使用 fit()、predict()、score()等函数进行训练、预测和评价。每个数据挖掘算法对应一个模型，记为 model，sklearn 库为每个模型提供的常用接口如表 11.1 所示。

表 11.1：sklearn 库为每个数据挖掘模型提供的统一接口

接口	用途
model.fit()	训练数据，监督模型时 fit(X, Y)，非监督模型时 fit(X)

model.predict()	预测测试样本
model.predict_proba()	输出预测结果相对应的置信概率
model.score()	用于评价模型在新数据上拟合质量的评分
model.transform()	对特征进行转换

11.4.2 sklearn 库解析：聚类

聚类是一个无监督学习过程，不需要进行样本数据训练。sklearn 提供了多种聚类函数供不同聚类目的使用，实例 21 中的 KMeans 是聚类中最为常用的算法之一，它属于基于划分的聚类方法。

KMeans 基本用法如下。

```
from sklearn.cluster import KMeans
model= KMeans()  #输入参数建立模型
model.fit(Data)     #将数据集 Data 提供给模型进行聚类
```

此外，还有基于层次的聚类方法，该方法将数据对象组成一棵聚类树，采用自底向上或自顶向下方式遍历，最终形成聚类。例如，sklearn 中的 AgglomerativeClustering 方法是一种聚合式层次聚类方法，其层次过程方向是自底向上。它首先将样本集合中的每个对象作为一个初始簇，然后将距离最近的两个簇合并组成新的簇，再将这个新簇与剩余簇中最近的合并，这种合并过程需要反复进行，直到所有的对象最终被聚到一个簇中。

AgglomerativeClustering 使用方法如下：

```
from sklearn.cluster import AgglomerativeClustering
model = AgglomerativeClustering()#输入参数建立模型
model.fit(Data)       #将数据集 Data 提供给模型进行聚类
```

DBSCAN 是一个基于密度的聚类算法。它不是基于距离而是密度进行分类，其目标是寻找被低密度区域分离的高密度区域，简单说，它把扎堆的点找出来，而点稀疏的区域作为分割区域。这种方法对噪声点的容忍性非常好，应用广泛。

DBSCAN 使用方法如下：

```
from sklearn.cluster import DBSCAN
model = DBSCAN()      #输入参数建立模型
model.fit(Data)       #将数据集 Data 提供给模型进行聚类
```

关于聚类，建议读者掌握 KMeans 方法。

微实例 11.1：10 个点的聚类。

假设有 10 个点: (1,2),(2,5),(3,4),(4,5),(5,8),(10,13),(11,10),(12,11),(13,15),(15,14)，请将它们分成 2 类，并绘制聚类效果。采用 KMeans 方法，代码如下。

微实例 11.1　　m11.1Cluster10Points.py

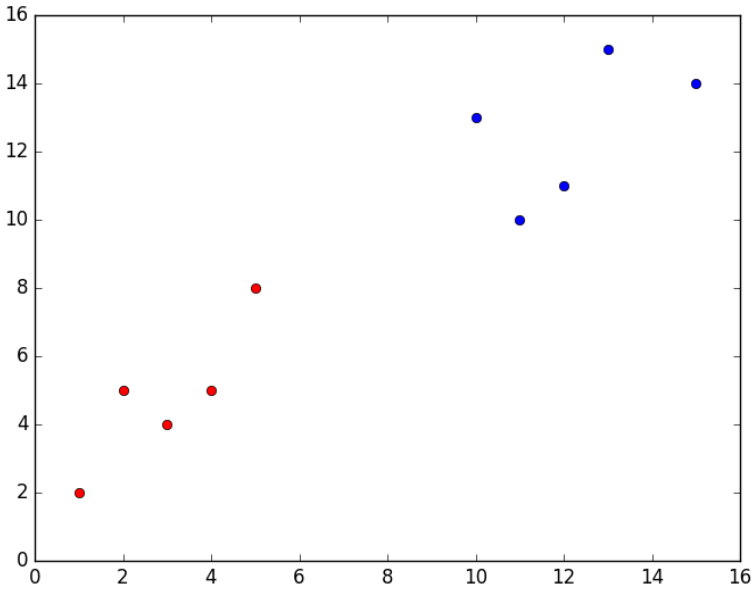
```
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt
dataSet = np.array([[1,2],[2,5],[3,4],[4,5],[5,8], \
                    [10,13],[11,10],[12,11],[13,15],[15,14]])
km = KMeans(n_clusters=2)
km.fit(dataSet)
plt.figure(facecolor = 'w')
plt.axis([0,16,0,16])
mark = ['or', 'ob']
for i in range(dataSet.shape[0]):
```

```
    plt.plot(dataSet[i, 0], dataSet[i, 1], mark[km.labels_[i]])
plt.show()
```

运行后的聚类结果如图 11.4 所示，结果如下：

类 A　(1,2),(2,5),(3,4),(4,5),(5,8)，

类 B　(10,13),(11,10),(12,11),(13,15),(15,14)



[图 11.4：微实例 11.1 的聚类结果]

11.4.3 sklearn 库解析：分类

很多应用需要一个能够智能分类的工具。类似人的思维过程，为了能够让程序学会分类，需要让程序学习一定带有标签的数据，建立数据和分类结果的关联，然后可以应用程序学到的“知识”分类未带标签数据的类别结果。与聚类不同，分类需要利用标签数据，分类问题是有监督学习问题。

最常用的分类算法是 K 近邻算法，该算法也是最简单的机器学习分类算法，对大多数问题都非常有效。K 邻近算法的主要思想是：如果一个样本在特征空间中最

相似(即特征空间中最邻近)的 K 个样本大多数属于某一个类别，则该样本也属于这个类别。K 邻近算法在 sklearn 库中的基本用法如下。

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier() #建立分类器模型
model.fit(Data,y)  #为模型提供学习数据 Data 和数据对应的标签结果 y
```

此外，决策树算法也是用于分类的数据挖掘经典算法之一，常用于特征含有类别信息的分类或回归问题，这种方法非常适合多分类情况。决策树算法的基本用法如下。

```
from sklearn.neighbors import DecisionTreeClassifier
model = DecisionTreeClassifier() #建立分类器模型
model.fit(Data,y)  #为模型提供学习数据 Data 和数据对应的标签结果 y
```

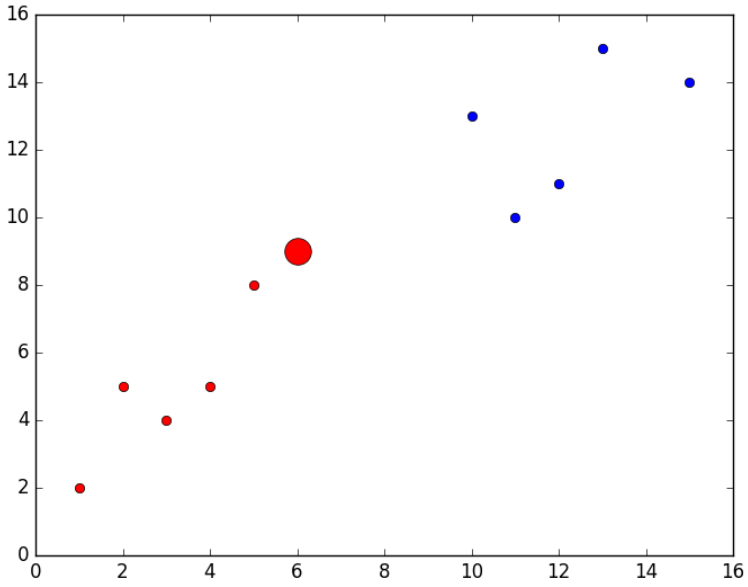
微实例 11.2：基于聚类结果的坐标点分类器。

微实例 11.1 两 10 个点分成了 2 类 A 和 B。现在有一个新的点（6，9），在分类结果 A 和 B 的基础上，新的点属于哪一类呢？采用 K 临近方法的分类代码如下，分类结果如图 11.5 所示。

微实例 11.2　　m11.2Classifier.py

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt
dataSet = np.array([[1,2],[2,5],[3,4],[4,5],[5,8], \
                    [10,13],[11,10],[12,11],[13,15],[15,14]])
```

```
km = KMeans(n_clusters=2)
km.fit(dataSet)
labels = km.labels_  #使用 KMeans 聚类结果进行分类
knn = KNeighborsClassifier()
knn.fit(dataSet,labels) #学习分类结果
data_new = np.array([[6,9]])
label_new = knn.predict(data_new) #对点(6,9)进行分类
plt.figure(facecolor = 'w')
plt.axis([0,16,0,16])
mark = ['or', 'ob']
for i in range(dataSet.shape[0]):
    plt.plot(dataSet[i, 0], dataSet[i, 1], mark[labels[i]])
plt.plot(data_new[0,0],  data_new[0,1],  mark[label_new[0]],
markersize=17)
plt.show()
```



[图 11.5：微实例 11.2 的分类结果]

从图 11.5 可以看到，点（6，9）被分为 A 类。这种分类采用了聚类结果。然而，

分类本身并不一定使用聚类结果，聚类结果只是给出了数据点和类别的一种对应关系。只要分类器学习了某种对应关系，它就能够进行分类。

微实例 11.3：基于人工分类的坐标点分类器。

微实例 11.1 中的 10 个点，由人工指定它们的分类如下：（可能是随机指定）

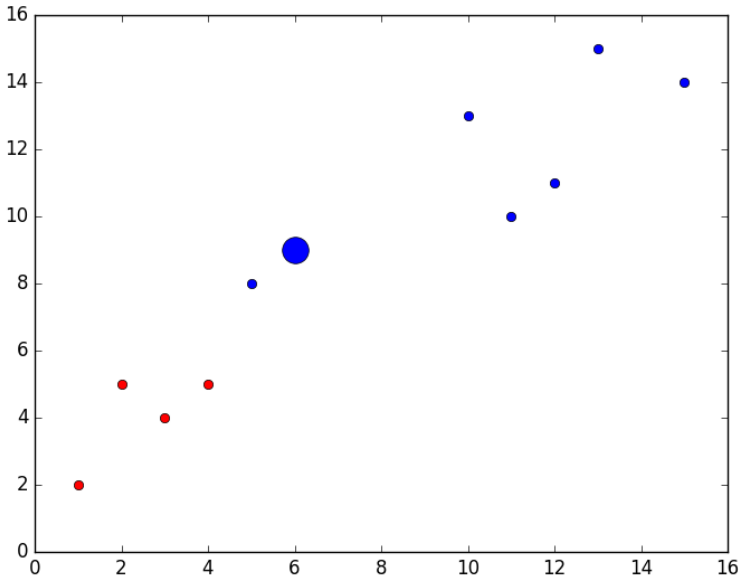
类 A　(1,2),(2,5),(3,4),(4,5)

类 B　(5,8),(10,13),(11,10),(12,11),(13,15),(15,14)

现在有一个新的点（6，9），在分类结果 A 和 B 的基础上，新的点属于哪一类呢？采用 K 临近方法的分类代码如下，分类结果如图 11.6 所示。

微实例 11.3　　m11.3Classifier.py

```
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
import matplotlib.pyplot as plt
dataSet = np.array([[1,2],[2,5],[3,4],[4,5],[5,8], \
                [10,13],[11,10],[12,11],[13,15],[15,14]])
labels = np.array([0,0,0,0,1,1,1,1,1,1])
knn = KNeighborsClassifier()
knn.fit(dataSet,labels) #学习分类结果
data_new = np.array([[6,9]])
label_new = knn.predict(data_new) #对点(6,9)进行分类
plt.figure(facecolor = 'w')
plt.axis([0,16,0,16])
mark = ['or', 'ob']
for i in range(dataSet.shape[0]):
   plt.plot(dataSet[i, 0], dataSet[i, 1], mark[labels[i]])
plt.plot(data_new[0,0],  data_new[0,1],  mark[label_new[0]],
markersize=17)
plt.show()
```



[图 11.6：微实例 11.3 的分类结果]

从图 11.6 可以看到，点（6，9）被分为 B 类。

11.4.4 sklearn 库解析：回归

回归是一个统计预测模型，用以描述和评估应变量与一个或多个自变量之间的关系，即自变量 X 与因变量 y 的关系。

最简单的回归模型是线性回归，它是数据挖掘中的基础算法之一。线性回归的思想是根据数据点形成一个回归函数 y=f(X)，函数的参数由数据点通过解方程获得。线性回归在 sklearn 库中的基本用法如下。

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()  #建立回归模型
model.fit(X,y) #建立回归模型，X 是自变量，y 是因变量
predicted = model.predict(X_new) #对新样本进行预测
```

很多实际问题都可以归结为逻辑回归问题，即回归函数的 y 值只有两个可能，也称为二元回归。逻辑回归可以使用 LogisticRegression()函数，接收数据并进行预测。逻辑回归在 sklearn 库中的基本用法如下。

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()  #建立回归模型
model.fit(X,y) #建立回归模型，X 是自变量，y 是因变量
predicted = model.predict(X_new) #对新样本进行预测
```

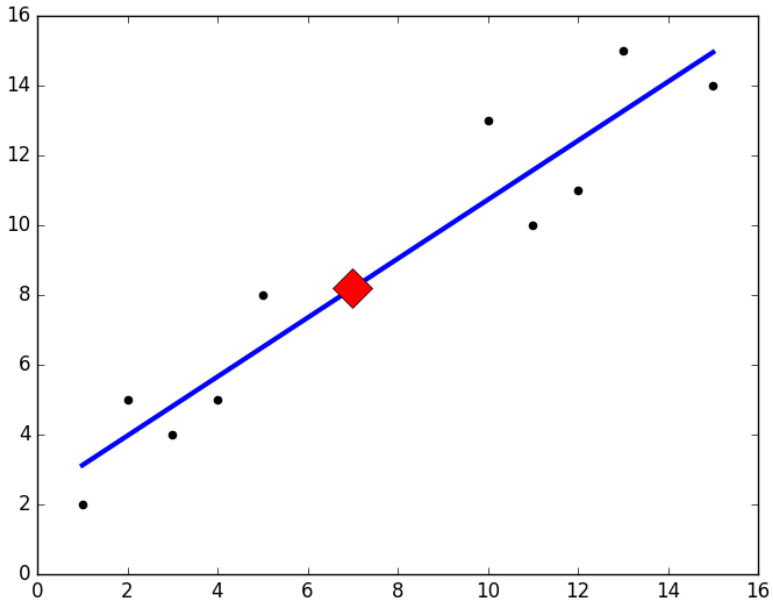
微实例 11.4：坐标点的预测器。

已知微实例 11.1 中的 10 个点，此时获得信息，将在横坐标 7 的位置出现一个新的点，却不知道纵坐标。请预测最有可能的纵坐标值。这是典型的预测问题，可以通过回归来实现。下面给出基于线性回归模型的预测器代码，预测结果如图 11.7 所示，预测点采用菱形标出。

微实例 11.4　　m11.4Regression.py

```
from sklearn import linear_model
import numpy as np
import matplotlib.pyplot as plt
dataSet = np.array([[1,2],[2,5],[3,4],[4,5],[5,8], \
                    [10,13],[11,10],[12,11],[13,15],[15,14]])
X = dataSet[:,0].reshape(-1,1)
y = dataSet[:,1]
linear = linear_model.LinearRegression()
linear.fit(X,y)  #根据横纵坐标构造回归函数
X_new = np.array([[7]])
plt.figure(facecolor = 'w')
plt.axis([0,16,0,16])
plt.scatter(X, y, color='black') #绘制所有点
```

```
13 plt.plot(X, linear.predict(X), color='blue',linewidth=3)
14 plt.plot(X_new , linear.predict(X_new ), 'Dr', markersize=17)
15 plt.show()
```



[图 11.7：微实例 11.4 的回归和预测结果]

思考与练习：

[E11.9]鸢尾花的花语是什么？

[E11.10]阐述聚类、分类和回归的含义和区别？

[E11.11]阐述聚类、分类和回归最常用的算法及算法思想。

11.5 实例 23：花辨识

要点：这是一个与鸢尾花相关的分类和预测实例。

在了解数据挖掘聚类、分类和回归方法的基础上，对 IRIS 数据集还可以进一步操作。IRIS 数据集中每个数据有四个属性特征：萼片长度、萼片宽度、花瓣长度、花瓣宽度。可以利用这些特征训练一个分类模型，用来对不同品种鸢尾花进行分类。

这个训练后的分类模型将能够根据上述 4 个属性辨识一朵花是否是鸢尾花以及它属于哪个品种。这样的模型通常被称为分类器，它能够体现一定的机器智能。

11.5.1 鸢尾花分类

分类模型中最常用的是 K 邻近算法，简称 KNN 算法。该算法首先需要学习，所以，将 IRIS 数据集随机分成 140 个数据的训练集和 10 个数据的测试集，并对预测准确率进行计算。由于 IRIS 数据集包括人工识别的标签，所以 140 个数据学习将比较准确。对于实际应用，可以采集一个小规模数据集并人工分类，再利用分类结果识别大数据集内容。

花辨识程序首先从 datasets 中导入数据，并将特征数据集和分类标签数据集存入 Data 变量和 Label 变量。使用 numpy 的随机列表生成函数，得到 150 个数据的随机排列数组 indices。利用 indices 列表将 IRIS 数据集随机分为包含 140 条数据的训练集和包含 10 条数据的测试集。

其次，为了计算预测的准确度，定义一个函数 calPrecision()，通过比较测试集数据的预测结果和 IRIS 中记录的真实分类情况的差异，对 KNN 分类器的准确度进行评价。依次对两个列表中相同位置的值进行比较，统计正确预测的次数。最后将正确次数除以总预测数，得到预测准确度。

使用 sklearn 库中的 KNN 分类器 KNeighborsClassifier，将分类器实例化并赋给变量 knn，调用 fit()函数将训练数据导入分类器进行训练，最后使用 predict()函数对测试集进行预测并将结果存入变量 predict_label。

拓展：训练集和测试集

数据挖掘中的分类问题求解一般采用机器学习类算法，这些算法通过学习一些数据集的特征属性并将其应用于新的数据。这需要使用算法时要把数据分成两部分。一部分称之为训练集，用以学习数据的特征属性，这部分数据需要人工标定，为数据生成标签。另一部分称为测试集，测试集不需要人工处理。

鸢尾花分类的全部程序如下。

实例 23.1　　e23.1IrisClassifier.py

```
#e23.1IrisClassifier.py
from sklearn import datasets
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
def loadIris():
    iris = datasets.load_iris()
    Data = iris.data
    Label = iris.target
    np.random.seed(0) #设随机种子
    indices = np.random.permutation(len(Data))
    DataTrain = Data[indices[:-10]]
    LabelTrain = Label[indices[:-10]]
    DataTest = Data[indices[-10:]]
    LabelTest = Label[indices[-10:]]
    return DataTrain, LabelTrain, DataTest, LabelTest
def calPrecision(prediction, truth):
    numSamples = len(prediction)
    numCorrect = 0
    for k in range(0, numSamples):
        if prediction[k] == truth[k]:
            numCorrect += 1
    precision = float(numCorrect) / float(numSamples)
    return precision
def main():
    iris_data_train, iris_label_train, iris_data_test\
                    ,iris_label_test=loadIris()
    knn = KNeighborsClassifier()
    knn.fit(iris_data_train, iris_label_train)
```

```
    predict_label = knn.predict(iris_data_test)
    print('测试集中鸢尾花的预测类别：{}'.format(predict_label))
    print('测试集中鸢尾花的真实类别：{}'.format(iris_label_test))
    precision  =  calPrecision(predict_label,iris_label_test)
print('KNN 分类器的精确度：  {}  %'.format(precision*100))
main()
```

程序运行效果如下。

```
>>>
测试集中鸢尾花的预测类别：[1 2 1 0 0 0 2 1 2 0]
测试集中鸢尾花的真实类别：[1 1 1 0 0 0 2 1 2 0]
KNN 分类器的精确度：  90.0 %
```

11.5.2 萼片宽度预测

IRIS 数据集已经给出了三种鸢尾花的花瓣数据，如果将目光投向外太空，在遥远的某个星球也生长着大片鸢尾花，只不过这个星球的植物都无比高大，所以它们的鸢尾花也非常高大。如果给出外星鸢尾花的萼片长度，希望预测它的萼片宽度，该怎么做呢？这就需要用到回归模型。

假设外星鸢尾花都是 setosa 类型，首先使用地球上 IRIS 数据集中所有 setosa 花型数据进行训练，得到线性回归模型。然后输入外星鸢尾花的萼片长度，使用线性回归拟合得到外星萼片宽度预测值。

实例代码直接使用了 sklearn 库的线性回归函数 LinearRegression，对 IRIS 模型中已知的 50 个 setosa 品种鸢尾花进行数据处理，得到回归模型。然后根据用户输入的外星鸢尾花萼片长度，预测对应的萼片宽度。

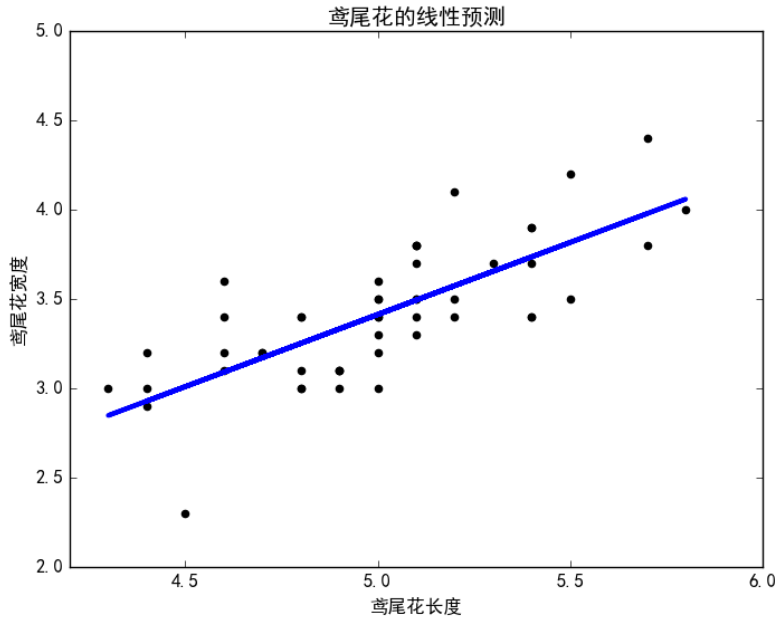
实例代码 23.2 首先定义一个载入数据的函数 loadIris()，它从 sklearn 库中加载 IRIS 数据集并获取 setosa 类型品种对应的数据，返回萼片长度 iris_X 和萼片宽度 iris_y；

然后，根据回归函数构造回归模型，对数据进行训练；最后，定义一个显示函数showLR()，将训练数据以散点图的形式绘制在图像中，同时绘制已经训练好的回归函数，如图 11.8 所示。实例代码 23.2 如下所示。

实例 23.2　　e23.2AlienIrisPredict.py

```
#e23.2AlienIrisPredict.py
from sklearn import datasets
from sklearn import linear_model
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams['font.family']='SimHei'
matplotlib.rcParams['font.sans-serif'] = ['SimHei']
def loadIris():
    iris = datasets.load_iris()
    iris_X = iris.data[:50, 0].reshape(-1, 1)
    iris_y = iris.data[:50, 1]
    return iris_X, iris_y
def showLR(regr, iris_X, iris_y):
    plt.figure(facecolor='w')
    plt.scatter(iris_X, iris_y,  color='black')
    plt.plot(iris_X,regr.predict(iris_X),color='blue',linewidth=3)
    plt.title('鸢尾花的线性预测')
    plt.xlabel('鸢尾花长度')
    plt.ylabel('鸢尾花宽度')
    plt.show()
def main():
    X,y= loadIris()
    liner = linear_model.LinearRegression()
    liner.fit(X,y)
    showLR(liner, X, y)
```

```
    x = input("请输入萼片长度(单位 cm)：  ")
    print("预测的萼片宽度是{:.2} cm".\
          format(liner.predict(float(x))[0]))
main()
```



[图 11.8：鸢尾花萼片宽度回归模型]

实例代码 23.2 执行后的结果如下，外星鸢尾花也能被预测了！

```
>>>
测试集中鸢尾花的预测类别：[1 2 1 0 0 0 2 1 2 0]
测试集中鸢尾花的真实类别：[1 1 1 0 0 0 2 1 2 0]
KNN 分类器的精确度：  90.0 %
```

思考与练习：

[E11.12]思考鸢尾花数据集中训练集和测试集的含义。

[E11.13]实例代码 23.1 中第 9 行代码执行后结果是什么？

[E11.14]说明花辨识实例代码 23.1 和实例代码 23.2 所引用第三方库的用途。

本章小结

本章阐述了数据挖掘的概念并介绍了实现数据挖掘的基本方法。利用 Skearn 库采用当代经典数据挖掘和机器学习算法进行聚类、分类和回归操作，通过给花朵找同类和鸢尾花辨识等实例展示数据挖掘的强大魅力。

程序练习题

[P11.1] 修改实例代码 22.1，在聚类结果输出坐标系中按照横坐标花瓣长度、纵坐标花瓣宽度方式显示。

[P11.2] 修改实例代码 23.1，将训练集数量改为 50 到 140，每 10 个数据一步，其余数据作为测试集，统计并分析 KNN 结果的正确率。

[P11.3] 第 12.4 节将介绍一个中国大学排名网站,请利用该网站对中国大学建立的评价标准及分数，构造一个数据集，使其符合被 sklearn 处理的格式。

[P11.4] 利用 P11.3 构建的数据集，对中国大学进行聚类，通过坐标系显示聚类结果，建议按照 4 个类别聚类。

[P11.5] 在 P11.4 基础上,如果类别数修改为 2 到 5,中国大学聚类会有什么变化？用坐标系显示聚类结果。

[P11.6] 编写根据国家统计局发布的某一城市过去 3 年房屋销售价格数据预测 2 年每季度的房价走势。