第 5 章

Scikit-learn 单机机器学习

Scikit-learn 是时下非常流行的基于 Python 语言的单机(单核/多核 CPU)机器学习工具。如图 5.1 所示,Scikit-learn 几乎涵盖了所有经典的机器学习算法,包括分类(Classification)、回归(Regression)、聚类(Clustering)、降维(Dimensionality Reduction)等。除此之外,Scikit-learn 还集成了特征提取、数据处理和模型评估等其他重要的机器学习相关功能。

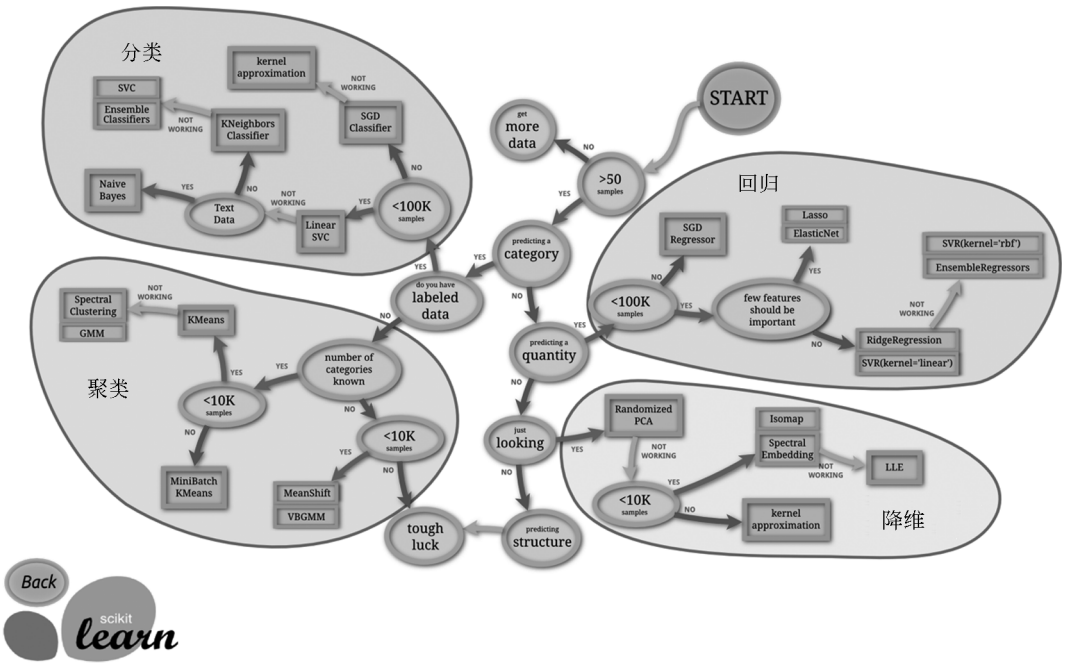


图 5.1　Scikit-learn 主要机器学习模型的全景图(图片引用自 Scikit-learn 官网)(见彩插)

Scikit-learn 在 2007 年成为 Google 夏季编程大赛的种子项目，并于 2010 年 2 月 1 日首次公开发布，至今已经历经了十余年的蓬勃发展。作为 Python 的重要机器学习程序库之一，Scikit-learn 的功能十分强大，内容全面，简单易用，而且支持跨平台运行，使得大量初学者都能够快速上手。

本章将尽可能全面地选取 Scikit-learn 中最为常用的单机机器学习功能，从一些经典的案例出发，向读者介绍足以理解并实践本书全部代码所需的 Scikit-learn 单机机器学习知识。

5.1　Scikit-learn 环境配置

为了本章的机器学习基础实践，我们创建一个新的虚拟环境，命名为 python_ml（Python Machine Learning 的缩写）。除了必要性地安装 NumPy、SciPy、pandas、Matplotlib 等程序库之外，这个虚拟环境中还需要搭建和配置 Scikit-learn，为后续基于 Python 进行机器学习实践奠定基础。

如图 5.2 所示，在 Windows 或者 macOS 系统中，可以直接使用 Anaconda Navigator，借助图形化的界面操作创建虚拟环境 python_ml；与此同时，指定新环境的 Python 解释器版本为 3.8。

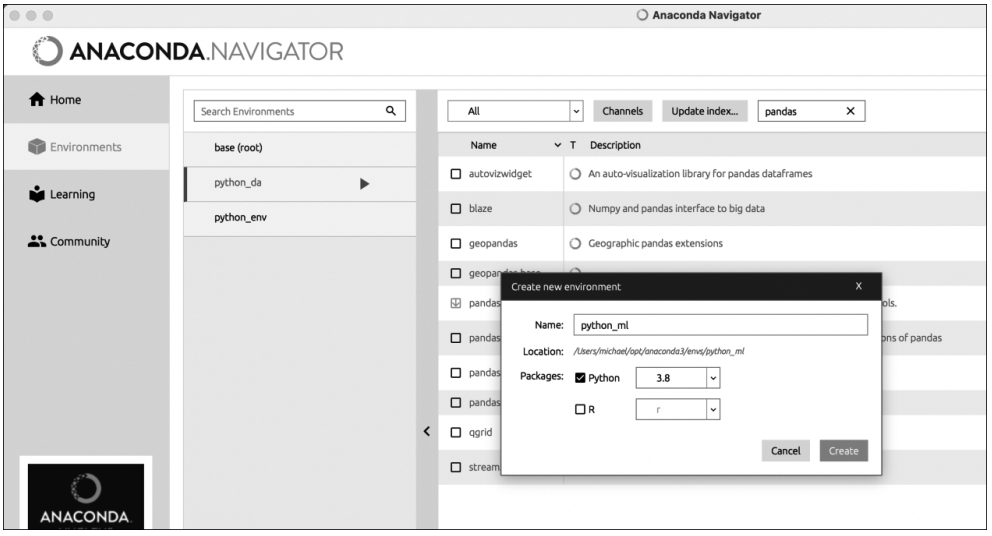


图 5.2　使用 Anaconda Navigator 创建用于机器学习的虚拟环境 python_ml，并使用 Python 3.8 作为解释器

另外，在 Windows、macOS，以及 Ubuntu 系统中，也可以通过在命令行/终端中输入命令 conda create -n python_ml python=3.8 创建新的虚拟环境。

虚拟环境 python_ml 创建好之后，我们可以在命令行/终端中使用命令 conda activate python_ml 切换到新的虚拟环境，并且如图 5.3 所示，在新虚拟环境的 Python 3.8 解释器中尝试导入 Scikit-learn 程序库[①]。运行的结果证明，作为一个新建的虚拟环境，其 Python 3.8 解释器并不会预装 Scikit-learn。

```
(base) michael@michaeldeMacBook-Pro ~ % conda activate python_ml
(python_ml) michael@michaeldeMacBook-Pro ~ % python
Python 3.8.11 (default, Aug  6 2021, 08:56:27)
[Clang 10.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import sklearn
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'sklearn'
>>> exit()
(python_ml) michael@michaeldeMacBook-Pro ~ %
```

图 5.3　虚拟环境 python_ml 的 Python 3.8 解释器没有预装 Scikit-learn

接下来，我们将分别演示如何使用 Anaconda Navigator 和 conda 命令在 Python 3.8 解释器中安装 Scikit-learn 程序库。

5.1.1　使用 Anaconda Navigator 搭建和配置环境

如图 5.4 所示，在 Anaconda Navigator 中首先切换到名称为 python_ml 的虚拟环境。然后，在右侧的程序库中搜索 scikit-learn，并且直接按照后续提示进行安装，即可配置好最新版本的 Scikit-learn 程序库。

5.1.2　使用 conda 命令搭建和配置环境

如图 5.5 所示，我们也可以在 Windows 的命令行或者 macOS/Ubuntu 系统的终端中，首先切换到名称为 python_ml 的虚拟环境，然后使用 conda 命令 conda install scikit-learn==0.24.2 自动安装和配置好版本号为 0.24.2 的 Scikit-learn 程序库[②]。

为了校验我们是否成功在虚拟环境 python_ml 的 Python 3.8 解释器中安装好了 Scikit-learn 程序库，可以在 Python 解释器中输入代码 import sklearn，尝试导入 Scikit-

① Scikit-learn 程序库在 Python 中的导入名称为 sklearn。

② 本书使用版本号为 0.24.2 的 Scikit-learn 程序库。

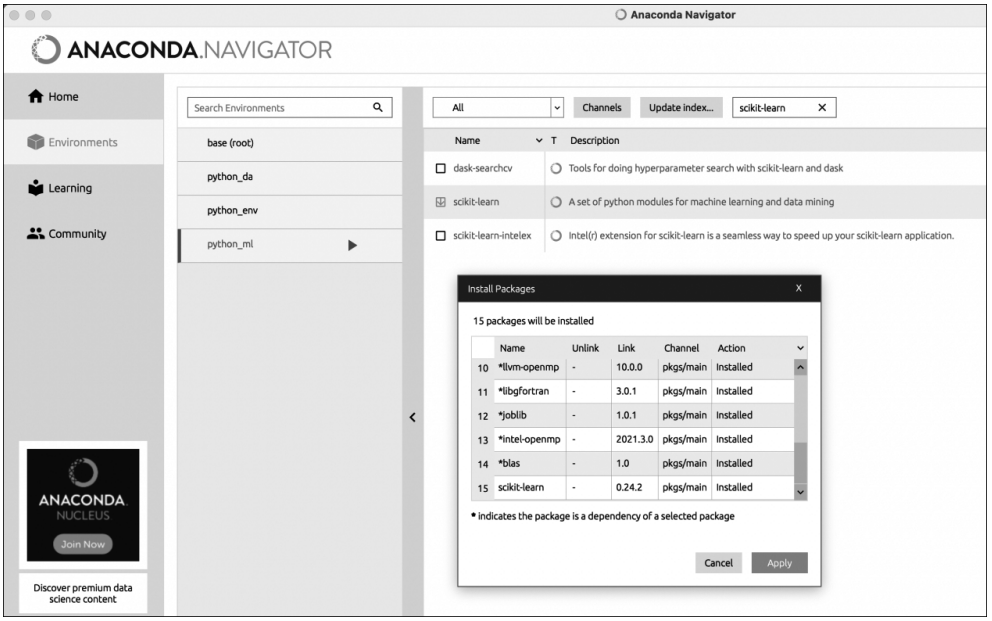


图 5.4　在虚拟环境 python_ml 中，使用 Anaconda Navigator 搭建和配置 Scikit-learn

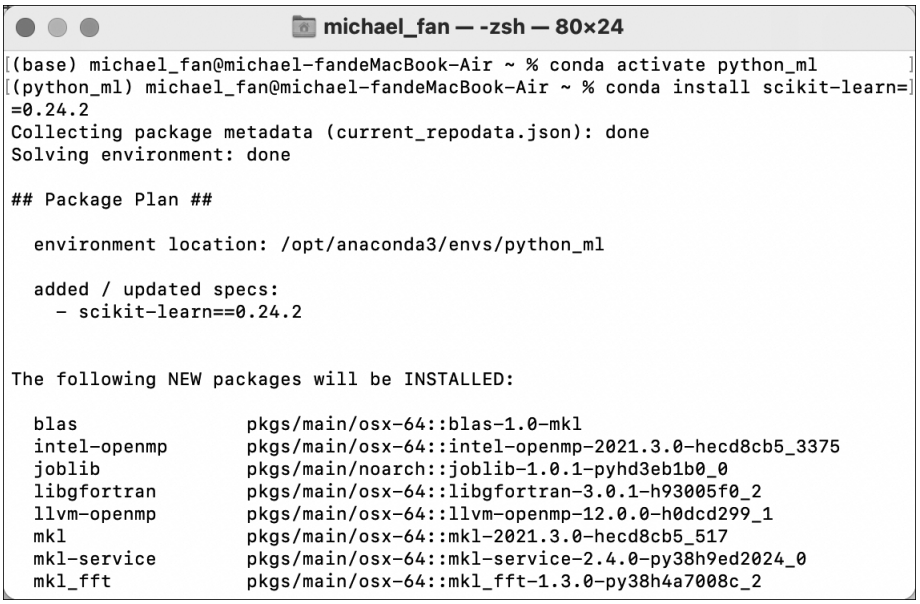


图 5.5　在虚拟环境 python_ml 中使用 conda 命令搭建和配置 Scikit-learn

learn 程序库。结果如图 5.6 所示，安装成功的 Scikit-learn 版本号为 0.24.2。

```
(python_ml) michael@michaeldeMacBook-Pro ~ % python
Python 3.8.11 (default, Aug  6 2021, 08:56:27)
[Clang 10.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import sklearn
>>> sklearn.__version__
'0.24.2'
>>> exit()
(python_ml) michael@michaeldeMacBook-Pro ~ %
```

图 5.6　在虚拟环境 python_ml 的 Python 3.8 解释器中尝试导入 Scikit-learn，验证环境搭建是否成功

5.2　Scikit-learn 无监督学习

无监督学习（Unsupervised Learning）着重于发现数据本身内在的分布特点，不需要对数据进行人工标注。

从功能的角度讲，无监督学习模型可以帮助我们对特征维度非常高的数据样本实施降维处理，保留那些最具有区分性的特征；同时，也使我们能够在三维空间中尽可能可视化地观察这些数据，甚至去发现数据的聚类效果。

本节采用经典的 iris（鸢尾花）数据集探讨和实践 Scikit-learn 中有关无监督学习的各项能力，包括数据特征降维、数据可视化，以及数据聚类等。

如图 5.7 所示，iris 数据集收录了 3 种常见的鸢尾品种，每种 50 朵，共 150 朵花的样本数据。其中，每一朵鸢尾被记录了 4 个维度的特征，分别是花瓣的长和宽，以及花萼的长和宽。

5.2.1　降维学习与可视化

降维学习指采用某种映射方法，将原本在高维（大于 3 维）空间中的数据点映射到低维度，甚至是可观察（小于 3 维）的空间中。降维学习的本质是学习一个映射函数 $f(x)=y$，其中，x 是原始数据点的向量表达形式；y 是数据点映射后的低维向量表达。我们之所以使用降维后的数据表示，是因为在原始数据的高维空间中，极有可能包含冗余的特征或者噪声信息。

（1）特征冗余指一些特征之间具有非常强的相关性，即知道其中一种特征的变化趋势，就可以立刻知道其他特征的变化（如正方形的边长与面积、周长都呈现正相关）。只要保留其中一种特征，就能在保证原有数据分布和信息的情况下有效简化数据，后面的模型

	sepalLength	sepalWidth	petalLength	petalWidth	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

图 5.7　iris 数据集的样本数量、特征名称以及类别标签情况

学习也会因此减少大量时间和空间开销。降维的最终目标是保证各个特征维度之间线性无关。

（2）噪声会为机器学习模型的训练带来不必要的误差，减弱模型最终的学习效果。因此，我们希望能够通过降维减少冗余信息和噪声带来的误差，同时也希望借助可视化能力来寻找数据内部的本质结构特征。

1. 主成分分析

主成分分析，顾名思义，就是找出数据中最主要的方面，用数据中最主要的方面来代替原始数据。主成分分析（Principal Component Analysis，PCA）是一种统计方法，它通过正交变换将一组可能存在相关性的数据转换为一组线性不相关的特征，转换后的这组变量叫主成分。这样一来，多组可能存在相关性的数据被压缩到维度更小，但是更具有区分度的特征上。所以，主成分分析是最重要的降维方法之一，在数据压缩消除冗余和数据噪声消除等领域都有着十分广泛的应用。

代码 5.1 展示了如何使用 Scikit-learn 的主成分分析算法，将 iris 数据集的四维数据特征降低（压缩）到二维，并使用 Matplotlib 进行数据可视化。

代码 5.1　使用 Scikit-learn 的主成分分析对数据特征进行降维

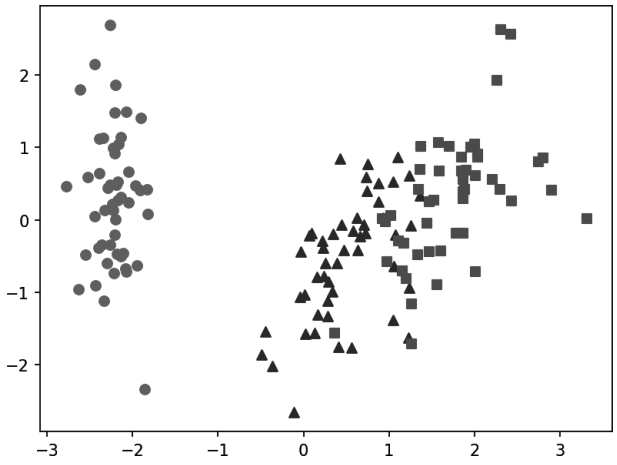
```
In[*]: from sklearn.datasets import load_iris


       #读取 iris 数据集
       X, y=load_iris(return_X_y=True)
```

```
In[*]: from sklearn.preprocessing import StandardScaler


       #初始化数据标准化处理器
       ss=StandardScaler()

       #标准化数据特征
       X=ss.fit_transform(X)
```

```
In[*]: from sklearn.decomposition import PCA


       #初始化主成分分析器，设定降维维度为 2
       pca=PCA(n_components=2)

       #对数据特征进行降维处理
       X=pca.fit_transform(X)
```

```
In[*]: from matplotlib import pyplot as plt


       colors = ['red', 'blue', 'green']
       markers = ['o', '^', 's']

       plt.figure(dpi=150)

       #可视化降维的数据
       for i in range(len(X)):
           plt.scatter(X[i, 0], X[i, 1], c=colors[y[i]], marker=markers[y[i]])

       plt.show()
```

Out[*]:



2. Isomap 算法

Isomap 算法曾发表在超一流期刊 Science 上。作为一种降维算法，它的核心在于发现并利用流形空间的特点，引入测地线距离，提出对应的距离计算方法。如图 5.8 所示，一个流形空间就像一块卷起来的布，对这个流形空间的降维就像是将这块布展开到一个

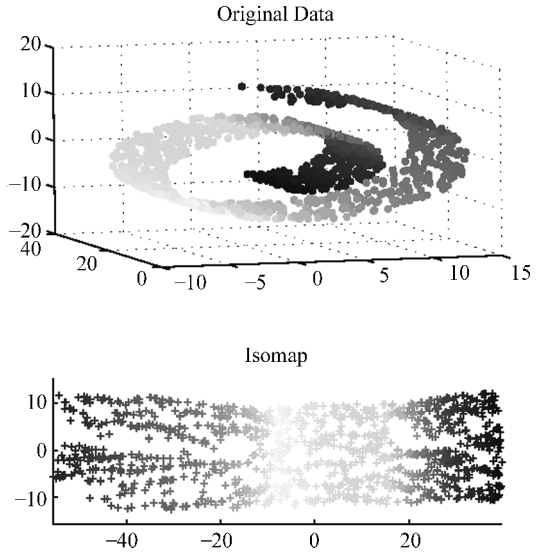


图 5.8　Isomap 算法解决三维流形分布数据的二维降维问题（见彩插）

二维平面，同时我们希望展开后的布能够在局部保持布的结构特征，也就是将其展开的过程，具体而言，就像两个人将其拉开一样。

在 Isomap 算法中引入了一个测地线的概念。在距离度量定义时，测地线可以定义为空间中两点的局域最短路径。形象地讲，在一个球面上，两点之间的测地线就是经过这两个点的大圆的弧线。对于非线性流形，Isomap 算法则是通过构建邻接图，利用图上的最短距离来近似测地线。Isomap 算法在降维后，希望保持样本之间的测地距离而不是欧氏距离，因为测地距离更能反映样本之间在流形中的真实距离。

代码 5.2 展示了如何使用 Scikit-learn 的 Isomap 算法将 iris 数据集的四维数据特征降低（压缩）到二维，并使用 Matplotlib 进行数据可视化。

代码 5.2　使用 Scikit-learn 的 Isomap 算法对数据特征进行降维

```
In[*]:  from sklearn.datasets import load_iris


        #读取 iris 数据集
        X, y=load_iris(return_X_y=True)
```

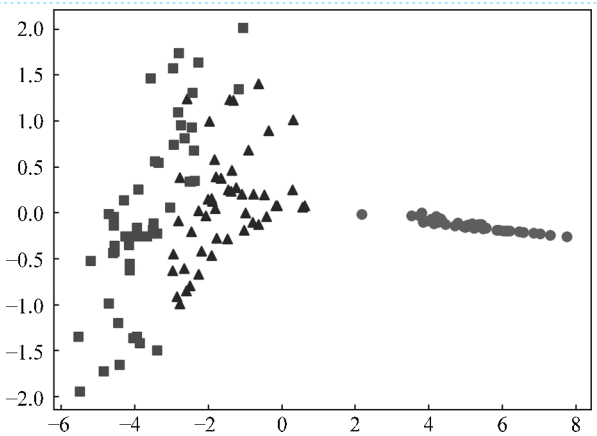
```
In[*]:  from sklearn.preprocessing import StandardScaler


        #初始化数据标准化处理器
        ss=StandardScaler()

        #标准化数据特征
        X=ss.fit_transform(X)
```

```
In[*]:  from sklearn.manifold import Isomap


        #初始化 Isomap，设定降维维度为 2
        isomap=Isomap(n_components=2)

        #对数据特征进行降维处理
        X=isomap.fit_transform(X)
```

```
In[*]:  from matplotlib import pyplot as plt


        colors =['red', 'blue', 'green']
```

```
markers = ['o', '^', 's']

plt.figure(dpi=150)

#可视化降维的数据
for i in range(len(X)):
    plt.scatter(X[i, 0], X[i, 1], c=colors[y[i]], marker=markers[y[i]])

plt.show()
```

Out[*]:



3. 局部线性嵌入

局部线性嵌入(Locally Linear Embedding,LLE)算法与传统的 PCA 等关注样本方差的降维方法相比,更注重降维时保持样本局部的线性特征。由于 LLE 在降维时保持了样本的局部特征,它被广泛用于图像识别,高维数据可视化等领域。

保持样本局部结构特征的方法有很多种,不同的保持方法对应不同的流形算法。例如前文所述,Isomap 算法在降维后,希望保持样本之间的测地距离而不是欧氏距离,因为测地距离更能反映样本之间在流形中的真实距离。

但是等距映射算法有一个问题,就是它要找到所有样本全局的最优解,当数据量很大,样本维度很高时,计算非常耗时。鉴于这个问题,LLE 通过放弃所有样本全局最优的降维,只是通过保证局部最优来降维。同时,假设样本集在局部是满足线性关系的,进一步减少降维的计算量。

代码 5.3 展示了如何使用 Scikit-learn 的 LLE 算法,将 iris 数据集的四维数据特征降